SUSE Linux Enterprise Server

10 SP3

www.novell.com

October 06, 2010

Heartbeat

Heartbeat

Contents

About This Guide

Heartbeat is an open source clustering software for Linux. Heartbeat ensures high availability and manageability of critical network resources including data, applications, and services. It is a multinode clustering product that supports failover, failback, and migration (load balancing) of individually managed cluster resources.

This guide is intended for administrators given the task of building Linux clusters. Basic background information is provided to build up an understanding of the Heartbeat architecture. Setup, configuration, and maintenance of a Heartbeat cluster are covered in detail. Heartbeat offers a graphical user interface as well as many command line tools. Both approaches are covered in detail to help the administrators choose the appropriate tool matching their particular needs.

NOTE

This manual covers Heartbeat version 2 and higher. Whenever the authors use the term “Heartbeat”, they refer to Heartbeat 2, even when no explicit version number is given.

This guide contains the following:

Overview
: Before starting to install and configure your cluster, learn about the features Heartbeat offers. Familiarize yourself with the Heartbeat terminology and its basic concepts.

Installation and Setup
: Learn about hardware and software requirements that must be met before you can consider installing and running your own cluster. Perform a basic installation and configuration using the Heartbeat graphical user interface.

Setting Up a Simple Resource
: After you have completed the basic cluster configuration, check out a quick step-by-step instruction of how to configure an example resource. Choose from a GUI-based approach or a command line-driven one.

Configuring and Managing Cluster Resources
: Managing resources encompasses much more than just the initial configuration. Learn how to use the Heartbeat graphical user interface to configure and manage resources.

Manual Configuration of a Cluster
: Managing resources encompasses much more than just the initial configuration. Learn how to use the Heartbeat command line tools to configure and manage resources.

Managing a Cluster
: Heartbeat provides a comprehensive set of command line tools to assist you in managing your own cluster. Get to know the most important ones for your daily cluster management tasks.

Creating Resources
: In case you consider writing your own resource agents for Heartbeat or modifying existing ones, get some detailed background information about the different types of resource agents and how to create them.

Troubleshooting
: Managing your own cluster requires you to perform a certain amount of troubleshooting. Learn about the most common problems with Heartbeat and how to fix them.

Terminology
: Refer to this chapter for some basic Heartbeat terminology that helps you understand the Heartbeat fundamentals.

1 Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

2 Documentation Updates

Expect updates to this documentation in the near future while it is extended to match the capabilities of the software itself and to address more and more use cases. By using the User Comments feature described in Section 1, “Feedback” (page vi), let us know on which aspects of Heartbeat this guide should elaborate.

For the latest version of this documentation, see the SLES 10 SP3 documentation Web site at `http://www.novell.com/documentation/sles10`.

3 Documentation Conventions

The following typographical conventions are used in this manual:

- `/etc/passwd`: filenames and directory names
- *`placeholder`*: replace *`placeholder`* with the actual value
- `PATH`: the environment variable PATH
- `ls`, `--help`: commands, options, and parameters
- `user`: users or groups
- Alt, Alt + F1: a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File*, *File* > *Save As*: menu items, buttons
- ► **amd64 ipf:** This paragraph is only relevant for the specified architectures. The arrows mark the beginning and the end of the text block. ◄

  ► **ipseries s390 zseries:** This paragraph is only relevant for the specified architectures. The arrows mark the beginning and the end of the text block. ◄
- *Dancing Penguins* (Chapter *Penguins*, ↑Another Manual): This is a reference to a chapter in another manual.

1 Overview

Heartbeat is an open source server clustering system that ensures high availability and manageability of critical network resources including data, applications, and services. It is a multinode clustering product for Linux that supports failover, failback, and migration (load balancing) of individually managed cluster resources. Heartbeat is shipped with SUSE Linux Enterprise Server 10 and provides you with the means to make virtual machines (containing services) highly available.

This chapter introduces the main product features and benefits of Heartbeat. Find several exemplary scenarios for configuring clusters and learn about the components making up a Heartbeat version 2 cluster. The last section provides an overview of the Heartbeat architecture, describing the individual architecture layers and processes within the cluster.

1.1 Product Features

Heartbeat includes several important features to help you ensure and manage the availability of your network resources. These include:

- Support for Fibre Channel or iSCSI storage area networks
- Multi-node active cluster, containing up to 16 Linux servers. Any server in the cluster can restart resources (applications, services, IP addresses, and file systems) from a failed server in the cluster.

- A single point of administration through either a graphical Heartbeat tool or a command line tool. Both tools let you configure and monitor your Heartbeat cluster.
- The ability to tailor a cluster to the specific applications and hardware infrastructure that fit your organization.
- Dynamic assignment and reassignment of server storage on an as-needed basis.
- Time-dependent configuration enables resources to fail back to repaired nodes at specified times.
- Support for shared disk systems. Although shared disk systems are supported, they are not required.
- Support for cluster file systems like OCFS 2.
- Support for cluster-aware logical volume managers like EVMS.

1.2 Product Benefits

Heartbeat allows you to configure up to 16 Linux servers into a high-availability cluster, where resources can be dynamically switched or moved to any server in the cluster. Resources can be configured to automatically switch or be moved in the event of a resource server failure, or they can be moved manually to troubleshoot hardware or balance the workload.

Heartbeat provides high availability from commodity components. Lower costs are obtained through the consolidation of applications and operations onto a cluster. Heartbeat also allows you to centrally manage the complete cluster and to adjust resources to meet changing workload requirements (thus, manually "load balance" the cluster).

An equally important benefit is the potential reduction of unplanned service outages as well as planned outages for software and hardware maintenance and upgrades.

Reasons that you would want to implement a Heartbeat cluster include:

- Increased availability
- Improved performance

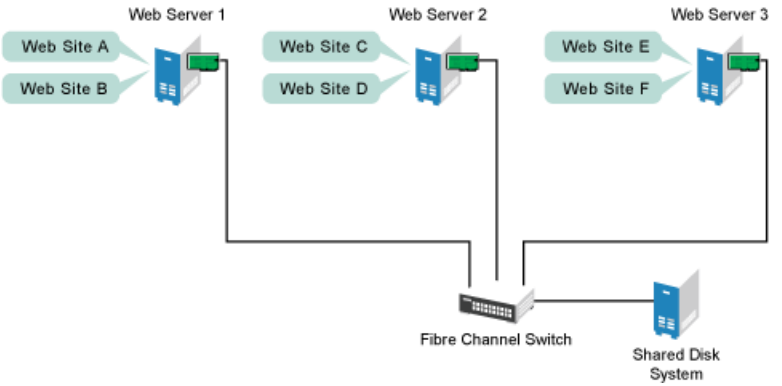
- Low cost of operation
- Scalability
- Disaster recovery
- Data protection
- Server consolidation
- Storage consolidation

Shared disk fault tolerance can be obtained by implementing RAID on the shared disk subsystem.

The following scenario illustrates some of the benefits Heartbeat can provide.

Suppose you have configured a three-server cluster, with a Web server installed on each of the three servers in the cluster. Each of the servers in the cluster hosts two Web sites. All the data, graphics, and Web page content for each Web site are stored on a shared disk subsystem connected to each of the servers in the cluster. The following figure depicts how this setup might look.

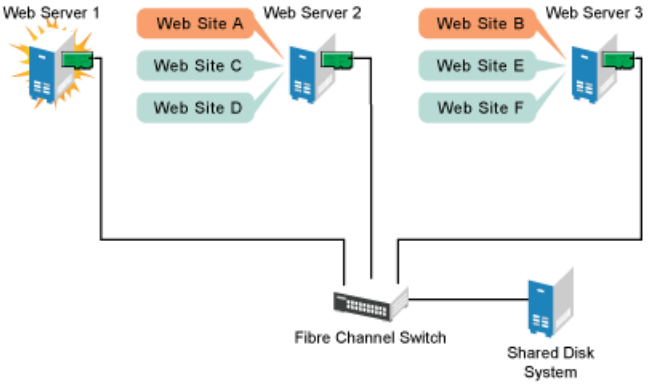
Figure 1.1 *Three-Server Cluster*



During normal cluster operation, each server is in constant communication with the other servers in the cluster and performs periodic polling of all registered resources to detect failure.

Suppose Web Server 1 experiences hardware or software problems and the users depending on Web Server 1 for Internet access, e-mail, and information lose their connections. The following figure shows how resources are moved when Web Server 1 fails.

Figure 1.2 *Three-Server Cluster after One Server Fails*



Web Site A moves to Web Server 2 and Web Site B moves to Web Server 3. IP addresses and certificates also move to Web Server 2 and Web Server 3.

When you configured the cluster, you decided where the Web sites hosted on each Web server would go should a failure occur. In the previous example, you configured Web Site A to move to Web Server 2 and Web Site B to move to Web Server 3. This way, the workload once handled by Web Server 1 continues to be available and is evenly distributed between any surviving cluster members.

When Web Server 1 failed, Heartbeat software

- Detected a failure
- Remounted the shared data directories that were formerly mounted on Web server 1 on Web Server 2 and Web Server 3.
- Restarted applications that were running on Web Server 1 on Web Server 2 and Web Server 3
- Transferred IP addresses to Web Server 2 and Web Server 3

In this example, the failover process happened quickly and users regained access to Web site information within seconds, and in most cases, without needing to log in again.

Now suppose the problems with Web Server 1 are resolved, and Web Server 1 is returned to a normal operating state. Web Site A and Web Site B can either automatically fail back (move back) to Web Server 1, or they can stay where they are. This is dependent on how you configured the resources for them. There are advantages and disadvantages to both alternatives. Migrating the services back to Web Server 1 will incur some downtime. Heartbeat also allows you to defer the migration until a period when it will cause little or no service interruption.

Heartbeat also provides resource migration capabilities. You can move applications, Web sites, etc. to other servers in your cluster without waiting for a server to fail.

For example, you could have manually moved Web Site A or Web Site B from Web Server 1 to either of the other servers in the cluster. You might want to do this to upgrade or perform scheduled maintenance on Web Server 1, or just to increase performance or accessibility of the Web sites.

1.3 Cluster Configurations

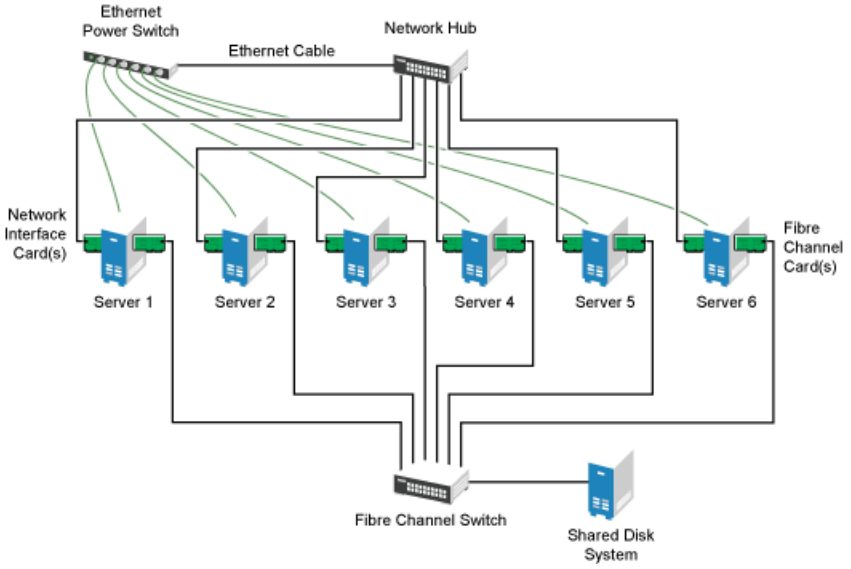
Heartbeat cluster configurations might or might not include a shared disk subsystem. The shared disk subsystem can be connected via high-speed Fibre Channel cards, cables, and switches, or it can be configured to use iSCSI. If a server fails, another designated server in the cluster automatically mounts the shared disk directories previously mounted on the failed server. This gives network users continuous access to the directories on the shared disk subsystem.

IMPORTANT: Shared Disk Subsystem with EVMS

When using a shared disk subsystem with EVMS, that subsystem must be connected to all servers in the cluster.

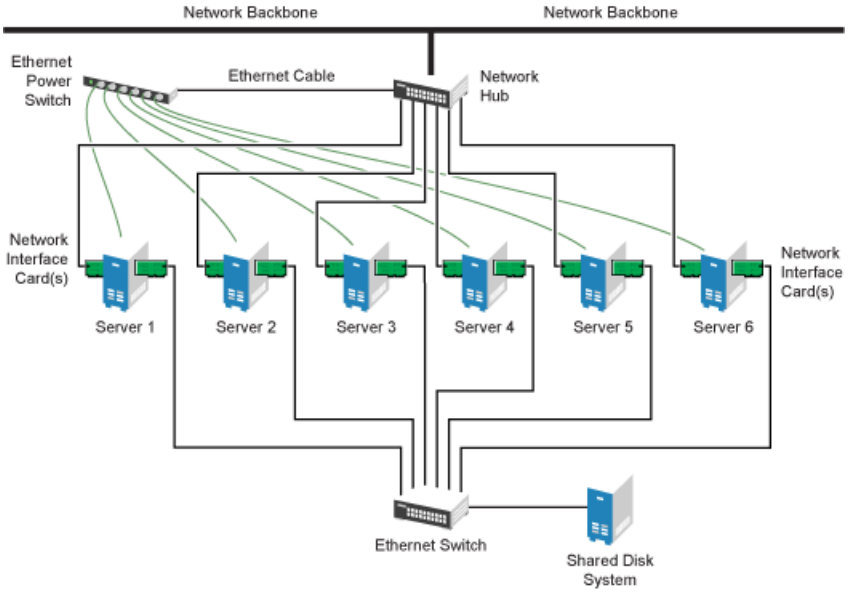
Typical Heartbeat resources might include data, applications, and services. The following figure shows how a typical Fibre Channel cluster configuration might look.

Figure 1.3 *Typical Fibre Channel Cluster Configuration*



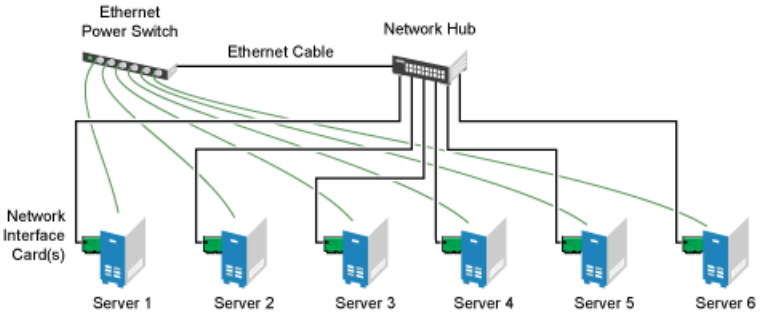
Although Fibre Channel provides the best performance, you can also configure your cluster to use iSCSI. iSCSI is an alternative to Fibre Channel that can be used to create a low-cost SAN. The following figure shows how a typical iSCSI cluster configuration might look.

Figure 1.4 *Typical iSCSI Cluster Configuration*



Although most clusters include a shared disk subsystem, it is also possible to create a Heartbeat cluster without a share disk subsystem. The following figure shows how a Heartbeat cluster without a shared disk subsystem might look.

Figure 1.5 *Typical Cluster Configuration Without Shared Storage*



1.4 Heartbeat Cluster Components

The following components make up a Heartbeat version 2 cluster:

- From 2 to 16 Linux servers, each containing at least one local disk device.
- Heartbeat software running on each Linux server in the cluster.
- Optional: A shared disk subsystem connected to all servers in the cluster.
- Optional: High-speed Fibre Channel cards, cables, and switch used to connect the servers to the shared disk subsystem.
- At least two communications mediums over which Heartbeat servers can communicate. These currently include Ethernet (mcast, ucast, or bcast) or Serial.
- A STONITH device. A STONITH device is a power switch which the cluster uses to reset nodes that are considered dead. Resetting non-heartbeating nodes is the only reliable way to ensure that no data corruption is performed by nodes that hang and only appear to be dead.

  See The High-Availability Linux Project [http://www.linux-ha.org/STONITH] for more information on STONITH.

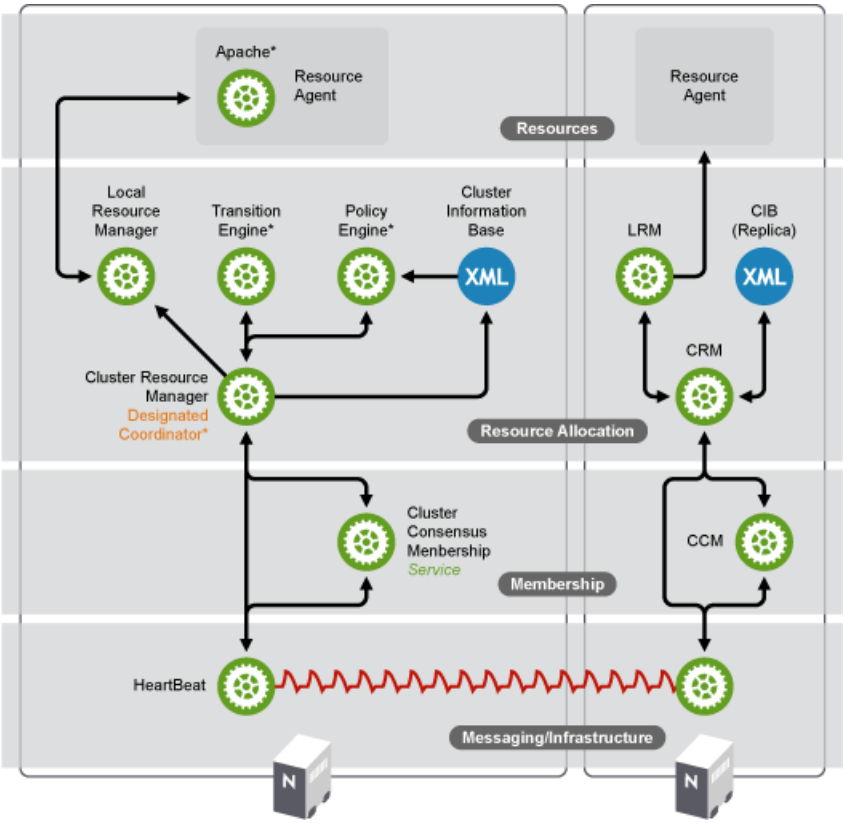
1.5 Architecture

This section provides a brief overview of the Heartbeat architecture. It identifies and provides information on the Heartbeat architectural components, and describes how those components interoperate.

1.5.1 Architecture Layers

Heartbeat has a layered architecture. Figure 1.6, “Heartbeat Architecture” (page 9) illustrates the different layers and their associated components.

Figure 1.6 *Heartbeat Architecture*



Messaging and Infrastructure Layer

The primary or first layer is the messaging/infrastructure layer, also known as the Heartbeat layer. This layer contains components that send out the Heartbeat messages containing “I'm alive” signals, as well as other information. The Heartbeat program resides in the messaging/infrastructure layer.

Membership Layer

The second layer is the membership layer. The membership layer is responsible for calculating the largest fully connected set of cluster nodes and synchronizing this view

to all of its members. It performs this task based on the information it gets from the Heartbeat layer. The logic that takes care of this task is contained in the Cluster Consensus Membership service, which provides an organized cluster topology overview (node-wise) to cluster components that are the higher layers.

Resource Allocation Layer

The third layer is the resource allocation layer. This layer is the most complex, and consists of the following components:

Cluster Resource Manager
: Every action taken in the resource allocation layer passes through the Cluster Resource Manager. If any other components of the resource allocation layer, or other components which are in a higher layer need to communicate, they do so through the local Cluster Resource Manager.

  On every node, the Cluster Resource Manager maintains the Cluster Information Base, or CIB (see Cluster Information Base (page 10) below). One Cluster Resource Manager in the cluster is elected as the Designated Coordinator (DC), meaning that it has the master CIB. All other CIBs in the cluster are a replicas of the master CIB. Normal read and write operations on the CIB are serialized through the master CIB. The DC is the only entity in the cluster that can decide that a cluster-wide change needs to be performed, such as fencing a node or moving resources around.

Cluster Information Base
: The Cluster Information Base or CIB is an in-memory XML representation of the entire cluster configuration and status, including node membership, resources, constraints, etc. There is one master CIB in the cluster, maintained by the DC. All the other nodes contain a CIB replica. If an administrator wants to manipulate the cluster's behavior, he can use either the `cibadmin` command line tool or the Heartbeat GUI tool.

NOTE: Usage of Heartbeat GUI Tool and `cibadmin`

The Heartbeat GUI tool can be used from any machine with a connection to the cluster. The `cibadmin` command must be used on a cluster node, and is not restricted to only the DC node.

Policy Engine (PE) and Transition Engine (TE)

Whenever the Designated Coordinator needs to make a cluster-wide change (react to a new CIB), the Policy Engine is used to calculate the next state of the cluster and the list of (resource) actions required to achieve it. The commands computed by the Policy Engine are then executed by the Transition Engine. The DC will send out messages to the relevant Cluster Resource Managers in the cluster, who then use their Local Resource Managers (see Local Resource Manager (LRM) (page 11) below) to perform the necessary resource manipulations. The PE/TE pair only runs on the DC node.

Local Resource Manager (LRM)

The Local Resource Manager calls the local Resource Agents (see Section "Resource Layer" (page 11) below) on behalf of the CRM. It can thus perform start / stop / monitor operations and report the result to the CRM. The LRM is the authoritative source for all resource related information on its local node.

Resource Layer

The fourth and highest layer is the Resource Layer. The Resource Layer includes one or more Resource Agents (RA). A Resource Agent is a program, usually a shell script, that has been written to start, stop, and monitor a certain kind of service (a resource). The most common Resource Agents are LSB init scripts. However, Heartbeat also supports the more flexible and powerful Open Clustering Framework Resource Agent API. The agents supplied with Heartbeat are written to OCF specifications. Resource Agents are called only by the Local Resource Manager. Third parties can include their own agents in a defined location in the file system and thus provide out-of-the-box cluster integration for their own software.

1.5.2 Process Flow

Many actions performed in the cluster will cause a cluster-wide change. These actions can include things like adding or removing a cluster resource or changing resource constraints. It is important to understand what happens in the cluster when you perform such an action.

For example, suppose you want to add a cluster IP address resource. To do this, you use either the `cibadmin` command line tool or the Heartbeat GUI tool to modify the master CIB. It is not required to use the `cibadmin` command or the GUI tool on the

Designated Coordinator. You can use either tool on any node in the cluster, and the local CIB will relay the requested changes to the Designated Coordinator. The Designated Coordinator will then replicate the CIB change to all cluster nodes and will start the transition procedure.

With help of the Policy Engine and the Transition Engine, the Designated Coordinator obtains a series of steps that need to be performed in the cluster, possibly on several nodes. The Designated Coordinator sends commands out via the messaging/infrastructure layer which are received by the other Cluster Resource Managers.

If necessary, the other Cluster Resource Managers use their Local Resource Manager to perform resource modifications and report back to the Designated Coordinator about the result. Once the Transition Engine on the Designated Coordinator concludes that all necessary operations are successfully performed in the cluster, the cluster will go back to the idle state and wait for further events.

If any operation was not carried out as planned, the Policy Engine is invoked again with the new information recorded in the CIB.

When a service or a node dies, the same thing happens. The Designated Coordinator is informed by the Cluster Consensus Membership service (in case of a node death) or by a Local Resource Manager (in case of a failed monitor operation). The Designated Coordinator determines that actions need to be taken in order to come to a new cluster state. The new cluster state will be represented by a new CIB.

2 Installation and Setup

A Heartbeat cluster can be installed and configured using YaST. During the Heartbeat installation, you are prompted for information that is necessary for Heartbeat to function properly. This chapter contains information to help you install, set up, and configure a Heartbeat cluster. Learn about the hardware and software requirements and which preparations to take before installing Heartbeat. Find detailed information about the installation process and the configuration of a STONITH device.

NOTE: Installing Heartbeat Software Packages on Cluster Nodes

This Heartbeat installation program does not copy the Heartbeat software package to cluster nodes. Prior to running the installation program, the Heartbeat software package must be installed on all nodes that will be part of your cluster. This can be done during the SUSE Linux Enterprise Server 10 installation, or after.

The Heartbeat installation program lets you create a new cluster or add nodes to an existing cluster. To add new nodes to an existing cluster, you must run the installation program from a node that is already in the cluster, not on a node that you want to add to the cluster.

After installing Heartbeat, you need to create and configure cluster resources. You might also need to create file systems on a shared disk (Storage Area Network, SAN) if they do not already exist and, if necessary, configure those file systems as Heartbeat cluster resources.

Both cluster-aware (OCFS 2) and non-cluster-aware file systems can be configured with Heartbeat. See the *Oracle Cluster File System 2 Administration Guide [http://www.novell.com/documentation/sles10/sles_admin/data/ocfs2.html]* for more information.

2.1 Hardware Requirements

The following list specifies hardware requirements for a Heartbeat cluster. These requirements represent the minimum hardware configuration. Additional hardware might be necessary, depending on how you intend to use your Heartbeat cluster.

- A minimum of two Linux servers. The servers do not require identical hardware (memory, disk space, etc.).
- At least two communication media (Ethernet, etc.) that allow cluster nodes to communicate with each other. The communication media should support a data rate of 100 Mbit/s or higher.
- A STONITH device.

2.2 Software Requirements

Ensure that the following software requirements are met:

- SLES 10 with all available online updates installed on all nodes that will be part of the Heartbeat cluster.
- The Heartbeat software package including all available online updates installed to all nodes that will be part of the Heartbeat cluster.

2.3 Shared Disk System Requirements

A shared disk system (Storage Area Network, or SAN) is recommended for your cluster if you want data to be highly available. If a shared disk subsystem is used, ensure the following:

- The shared disk system is properly set up and functional according to the manufacturer's instructions.
- The disks contained in the shared disk system should be configured to use mirroring or RAID to add fault tolerance to the shared disk system. Hardware-based RAID is recommended. Software-based RAID1 is not supported for all configurations.
- If you are using iSCSI for shared disk system access, ensure that you have properly configured iSCSI initiators and targets.

2.4 Preparations

Prior to installation, execute the following preparatory steps:

- Configure hostname resolution by either setting up a DNS server or by editing the `/etc/hosts` file on each server in the cluster as described in Procedure 2.1, "Modifying `/etc/hosts` with YaST" (page 15). It is essential that members of the cluster are able to find each other by name. If the names are not available, internal cluster communication will fail.
- Configure time synchronization by making Heartbeat nodes synchronize to a time server outside the cluster as described in Procedure 2.2, "Configuring the Heartbeat Servers for Time Synchronization" (page 16). The cluster nodes will use the time server as their time synchronization source.

  Heartbeat does not require that time is synchronized for each of the nodes in the cluster, however, it is highly recommended if you ever plan to look at logfiles.

Procedure 2.1 *Modifying /etc/hosts with YaST*

1. On one SLES 10 server, open YaST, and in left column, select *Network Services* > *Hostnames*.
2. In the *Host Configuration* window, click *Add* and fill in the necessary information in the pop-up window for one other server in the cluster.

   This information includes the IP address, hostname (for example node2.novell.com), and host alias (for example node2) of the other server.

Use node names for host aliases. You can find node names by executing `uname -n` at a command prompt on each server.

3 Click *OK*, and then repeat Step 2 (page 15) to add the other nodes in the cluster to the `/etc/hosts` file on this server.

4 Repeat Step 1 (page 15) through Step 3 (page 16) on each server in your Heartbeat cluster.

5 To check if hostname resolution is functioning properly, ping the node name (host alias) you specified in Step 2 (page 15) .

Procedure 2.2 *Configuring the Heartbeat Servers for Time Synchronization*

Time synchronization will not start unless the date and time on the cluster servers is already close. To manually set the date and time on cluster servers, use the `date` command at the command line of each cluster server to view, and if necessary change, the date and time for that server.

To configure the nodes in the cluster to use the time server as their time source proceed as follows:

1 On a cluster server, start YaST, click *Network Services*, then click *NTP Client*.

2 Choose to have the NTP daemon start during boot and then enter the IP address of the time server you configured.

3 Click *Finish* and reboot this server to ensure the service starts correctly.

4 Repeat Step 1 (page 16) through Step 3 (page 16) on the other non-time server nodes in the cluster.

5 Reboot all cluster nodes. After rebooting the nodes, time should be synchronized properly.

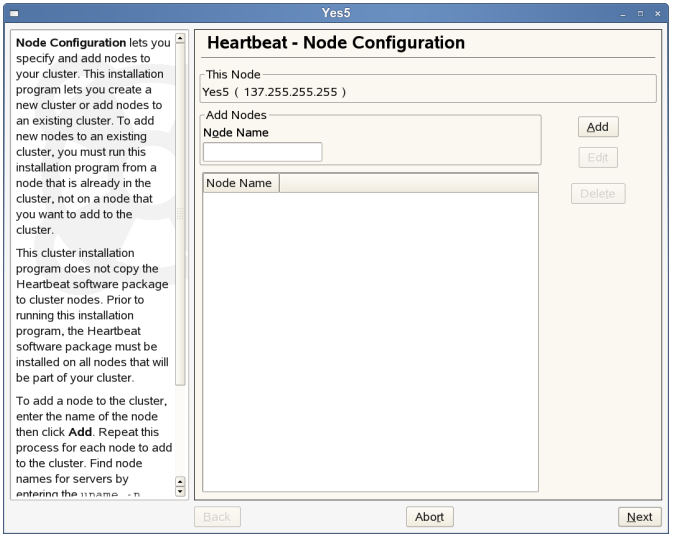
6 To check if time synchronization is functioning properly, run the `ntpq -p command` on each cluster node. Running the `ntpq -p` command on a non-time server node will display the server that is being used as the time source.

2.5 Installing Heartbeat

1 Start YaST and select *Miscellaneous* > *High Availability* or enter `yast2 heartbeat` to start the YaST Heartbeat module. It lets you create a new cluster or add new nodes to an existing cluster.

2 On the *Node Configuration* screen, add a node to the cluster by specifying the node name of the node you want to add, then click *Add*. Repeat this process for each node you want to add to the cluster, then click *Next*.

Figure 2.1 *Node Configuration*



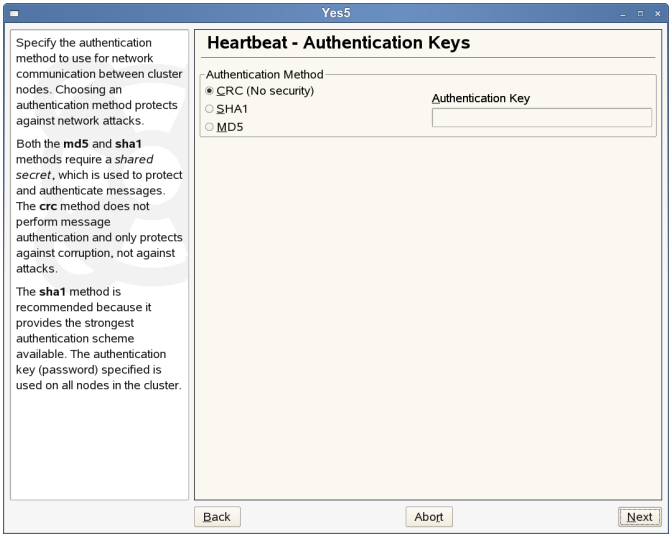
You can find node names for servers by entering `uname -n` on each node.

The installation program will not let you add this node, because the node name of this server is already automatically added to the cluster.

If after adding a node to the cluster, you need to specify a different node name for that node, double-click the node you want to edit, change the node name, then click *Edit*.

3 On the *Authentication Keys* screen, specify the authentication method the cluster will use for communication between cluster nodes, and if necessary an *Authentication Key* (password). Then click *Next*.

Figure 2.2 *Authentication Keys*



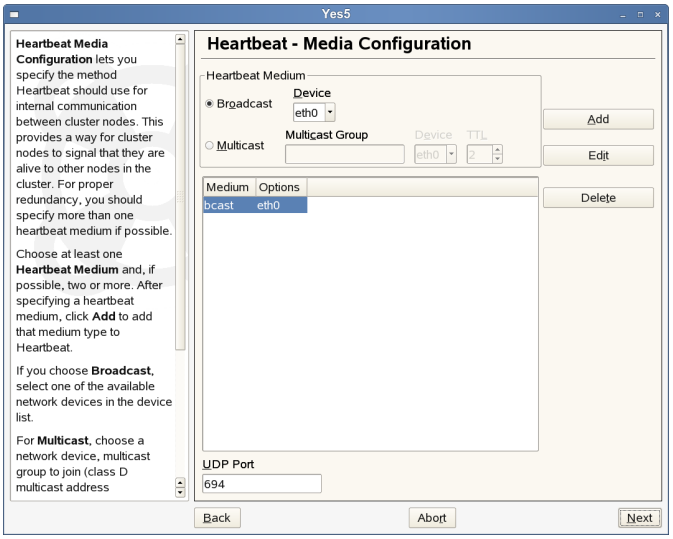
Both the MD5 and SHA1 methods require a shared secret, which is used to protect and authenticate messages. The CRC method does not perform message authentication, and only protects against corruption, not against attacks.

The SHA1 method is recommended, because it provides the strongest authentication scheme available. The authentication key (password) you specify will be used on all nodes in the cluster.

When running this installation program on the other cluster nodes, you should choose the same authentication method for all nodes.

4 On the *Media Configuration* screen, specify the method Heartbeat will use for internal communication between cluster nodes. This configuration is written to `/etc/ha.d/ha.cf`.

Figure 2.3 *Media Configuration*



This provides a way for cluster nodes to signal that they are alive to other nodes in the cluster. For proper redundancy, you should specify at least two Heartbeat media if possible. Choose at least one Heartbeat medium, and if possible, more than one:

- If you choose *Broadcast*, select one of the available network devices in the *Device* list.
- For *Multicast*, choose a network *Device*, *Multicast Group* to join (class D multicast address 224.0.0.0 - 239.255.255.255) and the *TTL* value (1-255).

UDP Port sets the UDP port that is used for the broadcast media. Leave this set to the default value (694) unless you are running multiple Heartbeat clusters on the same network segment, in which case you need to run each cluster on a different port number.

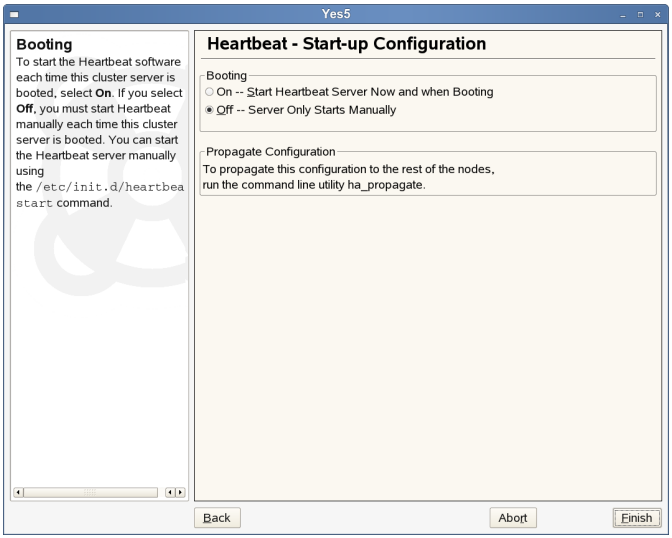
NOTE: UDP Port Settings

Note that the UDP port setting only apply to broadcast media, not to the other media you may use. When editing UDP ports manually in `/etc/ha.d/ha.cf`, the `udpport` entry must precede the `bcast` entry it belongs to, otherwise Heartbeat will ignore the port setting.

After specifying a Heartbeat medium, click *Add* to add that medium type to Heartbeat and proceed with *Next*.

5 On the *Start-up Configuration* screen, choose whether you want to start the Heartbeat software on this cluster server each time it is booted.

Figure 2.4 *Startup Configuration*



If you select *Off*, you must start Heartbeat manually each time this cluster server is booted. You can start the Heartbeat server manually using the `rcheartbeat start` command.

To start the Heartbeat server immediately, select *On - Start Heartbeat Server Now and when Booting*.

To start Heartbeat on the other servers in the cluster when they are booted, enter `chkconfig heartbeat on` at the server console of each of those servers. You can also enter `chkconfig heartbeat off` at the server console to have Heartbeat not start automatically when the server is rebooted.

6 To configure Heartbeat on the other nodes in the cluster run `/usr/lib/heartbeat/ha_propagate` on the heartbeat node you just configured. On 64-bit systems, the command `ha_propagate` is located below `/usr/lib64/heartbeat/`.

This will copy the Heartbeat configuration to the other nodes in the cluster. You will be prompted to enter the root user password for each cluster node.

7 Run `rcheartbeat start` to start heartbeat on each of the nodes where the configuration has been copied.

2.6 Configuring STONITH

STONITH is the service used by Heartbeat to protect shared data. Heartbeat is capable of controlling a number of network power switches, and can prevent a potentially faulty node from corrupting shared data by using STONITH to cut the power to that node.

With the STONITH service configured properly, Heartbeat does the following if a node fails:

- Notices that the node is not sending “I'm alive” packets out to the cluster.
- Sends out pre-stop notifications to the other cluster nodes.

  These notifications include messages that the failed node will be powered off.
- Instructs the STONITH service to power off the failed node.
- Sends post-stop notifications to other cluster nodes after successfully powering off the failed node.

  These notifications include messages that the failed node will be powered off.

For Heartbeat, STONITH must be configured as a cluster resource. After reviewing Chapter 4, *Configuring and Managing Cluster Resources* (page 29), continue with Procedure 2.3, "Configuring STONITH as a Cluster Resource" (page 22).

Procedure 2.3 *Configuring STONITH as a Cluster Resource*

1. Start the HA Management Client and log in to the cluster as described in Section 4.1, "Graphical HA Management Client" (page 29).

2. To generally enable the use of STONITH, select the *linux-ha* entry in the left pane and click the *Configurations* tab on the right.

3. Activate the *STONITH Enabled* check box.

4. From the menu, select *Resources* > *Add New Item* or click the + button.

5. Choose *Native* as the item type.

6. Specify the *Resource ID* (name) for the STONITH resource.

7. In the *Type* section of the window, scroll down and select the type of STONITH device that corresponds to your network power switch.

8. (Conditional) After selecting a STONITH resource type, a line for that resource type might be added to the *Parameters* section of the screen. If a line is added, click the line once and then click the line again under the *Value* heading to open a field where you can add the needed value.

   Some STONITH options require you to add parameter values in the parameters section of the page. For example, you may be required to add host names (server names) for each cluster node in your cluster. You can find the hostname for each server using the `uname -n` command on the server. You can add multiple hostnames in the provided field using a comma or a space to separate the names.

9. Select the *Clone* check box. This allows the resource to simultaneously run on multiple cluster nodes.

10. In the *clone_node_max* field, enter the number of instances of STONITH that will run on a given node. This value should normally be set to `1`.

11. In the *clone_max* field, enter the number of nodes in the cluster that will run the STONITH service. Be mindful of the number of concurrent connections your STONITH device supports.

12. Enter the *Clone or Master/Slave ID*.

13. Click *Add* to add the STONITH resource to the cluster. It now appears below the *Resources* entry in the left pane.

14. Select the resource in the left pane and click *Resource* > *Start* to start the STONITH resource.

After creating the STONITH resource, you must create location constraints for it. Location constraints determine which nodes STONITH can run on in the cluster. See Constraints [http://www.linux-ha.org/ClusterInformationBase/SimpleExamples] on the High Availability Linux Web site.

Starting the STONITH resource will cause it to start on the nodes that have been specified with its resource location constraints.

3 Setting Up a Simple Resource

Once your cluster is installed and set up as described in Chapter 2, *Installation and Setup* (page 13), you can start adding resources to your configuration. Configure resources either with the Heartbeat GUI or manually by using the command line tools.

In the following, find an example of how to configure an IP address as a resource either manually or with the Heartbeat GUI.

3.1 Configuring a Resource with the Heartbeat GUI

Creating a sample cluster resource and migrating it to another server can help you test to ensure your Heartbeat cluster is functioning properly. A simple resource to configure and migrate is an IP address.

Procedure 3.1 *Creating an IP Address Cluster Resource*

1. Start the HA Management Client and log in to the cluster as described in Section 4.1, “Graphical HA Management Client” (page 29).
2. Click *Resources*, then click *Add New Item*, or click the + button.
3. Choose *Native* as the resource item type, then click *OK*.

4 Enter a *Resource ID* (name) for the IP address resource. For example, `ipaddress1`.

5 In the *Type* section of the page, scroll down the list and select `IPaddr (OCF Resource Agent)` as the resource type.

6 In the *Parameters* section of the page, find the line that was added for the IP address resource, click the line once, then click the line again under the *Value* heading to open a text field.

7 Add the IP address for the IP address cluster resource.

8 Click the *Add Parameter* button and from the drop-down list, specify `nic` as *Name* and `eth0` as *Value*, then click *OK*.

The name and value are dependent on your hardware configuration and what you chose for the media configuration during the Heartbeat installation.

9 Click the *Add* button at the bottom of the page to add the resource to the cluster.

10 Select the resource in the left pane of the main window, then click *Resources* > *Start* to start the new resource on the cluster.

Procedure 3.2 *Migrating Resources to Another Node*

To migrate the newly created resource to another node in the cluster, you can use either the HA Management Client or the command line.

1 It the command line, use the following command:

```
crm_resource -M -r resource_name -H hostname
```

For example, to migrate a resource named `ipaddress1` to a cluster node named `node2`, you would need to enter:

```
crm_resource -M -r ipaddress1 -H node2
```

2 To use the HA Management Client for migration, select the resource you want to migrate in the left pane of the main window and select *Resources* > *Migrate Resource*.

3 In the new window, select `ipaddress1` as *Resource* to migrate and select `node1` from the *To Node* drop-down list.

3.2 Manual Configuration of a Resource

Resources are any type of service that a computer provides. Resources are known to Heartbeat when they may be controlled by RAs (Resource Agents), which are LSB scripts, OCF scripts, or legacy Heartbeat 1 resources. All resources are configured in the CIB (Cluster Information Base) in the `resources` section. For an overview of available resources, look at Appendix A, *HB OCF Agents* (page 109).

To add a resource to the current configuration, first write an XML file with the specific date for this resource. For example, to add the IP address `10.10.0.1` to your cluster, use the following example:

```
<primitive id="ip_1"❶
    class="ocf" ❷
    type="IPaddr" ❷
    provider="heartbeat" ❷ >
  <instance_attributes>
    <attributes>❸
      <nvpair name="ip" value="10.10.0.1"❹/>
    </attributes>
  </instance_attributes>
</primitive>
```

❶ The value of the `id` attribute of the `primitive` tag may be chosen freely. Like all IDs in XML, it must be unique and should follow a system. For example, `ip_1` may be read as the first `ip` primitive.

❷ The three attributes `class`, `type`, and `provider` determine the exact script that is used for this primitive. In this example, the script is at `/usr/lib/ocf/resource.d/heartbeat/IPaddr`.

❸ All the attributes for a resource agent are entered in a list of `nvpair` tags. This should not be confused with the XML attributes that are added, for example, to the `primitive` tag.

❹ In this example, the RA attribute `ip` is set to `10.10.0.1`. For the `IPaddr` RA, this RA attribute is mandatory, as can be seen in Appendix A, *HB OCF Agents* (page 109).

NOTE

When configuring a resource with Heartbeat, the same resource should not be initialized by init. Heartbeat should be responsible for all service start or stop actions.

To add this `IPaddr` configuration to the cluster, first save the configuration to a file named `ip_1.xml`. Add this file to the cluster configuration with the command:

```
cibadmin -o resources -C -x ip_1.xml
```

If the configuration was successful, a new resource appears in `crm_mon` that is started on a random node of your cluster.

4 Configuring and Managing Cluster Resources

Cluster resources must be created for every resource or application you run on servers in your cluster. Cluster resources can include Web sites, e-mail servers, databases, file systems, virtual machines, and any other server-based applications or services you want to make available to users at all times.

You can either use the graphical HA Management Client utility, or the `cibadmin` command line utility to create resources.

This chapter introduces the graphical HA Management Client and then covers several topics you need when configuring a cluster: creating resources, configuring constraints, specifying failover nodes and failback nodes, configuring resource monitoring, starting or removing resources, configuring resource groups or clone resources, and migrating resources manually.

4.1 Graphical HA Management Client

When starting the Heartbeat GUI, you need to connect to a cluster.

NOTE: Password for the `hacluster` User

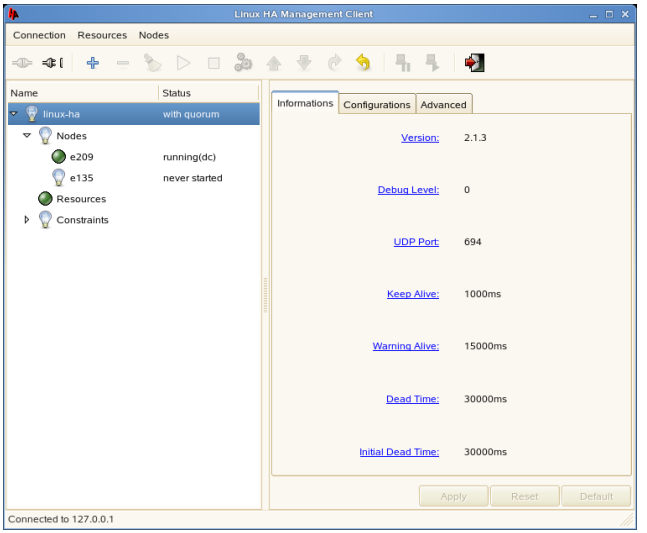
The Heartbeat installation creates a linux user named `hacluster`. Prior to using the HA Management Client, you must set the password for the `hacluster` user. To do this, enter `passwd hacluster` at the command line and enter a password for the `hacluster` user.

Do this on every node where you will run the HA Management Client utility.

To start the HA Management Client, enter `hb_gui` at the command line of a Linux server or workstation. Log in to the cluster by selecting *Connection* > *Login*. You are prompted for a username and password.

If you are running the HA Management Client on a Heartbeat cluster server, the server and username fields should already be filled in. If not, specify the server IP address of a cluster server and `hacluster` as the username.

After being connected, your main window shows the cluster components, resources and constraints:



Depending on which entry you select in the left pane, several tabs appear in the right pane of the main window. For example, if you select the topmost entry, *linux-ha*, you can access three tabs on the right in the main window, allowing you to view general *Information* on the cluster or to change certain options and aspects on the *Configurations* and *Advanced* tabs.

In the following, find some examples how to create and manage cluster resources with the HA Management Client.

4.2 Creating Cluster Resources

To add a cluster resource:

1. Start the HA Management Client and log in to the cluster as described in Section 4.1, “Graphical HA Management Client” (page 29).

2. Select *Resources* > *Add New Item*, or click the + button.

3. Choose the resource item type you want to create, then click *OK*.

   The possible options are:

   - Native
   - Group
   - Location (Constraint)
   - Order (Constraint)
   - Colocation (Constraint)

   All but the *Native* and *Group* resource item types are constraints. You must have a resource or group created in order to create a constraint and add that constraint to a resource.

4. Enter the Resource ID (name) and if desired, choose a group to add the resource to.

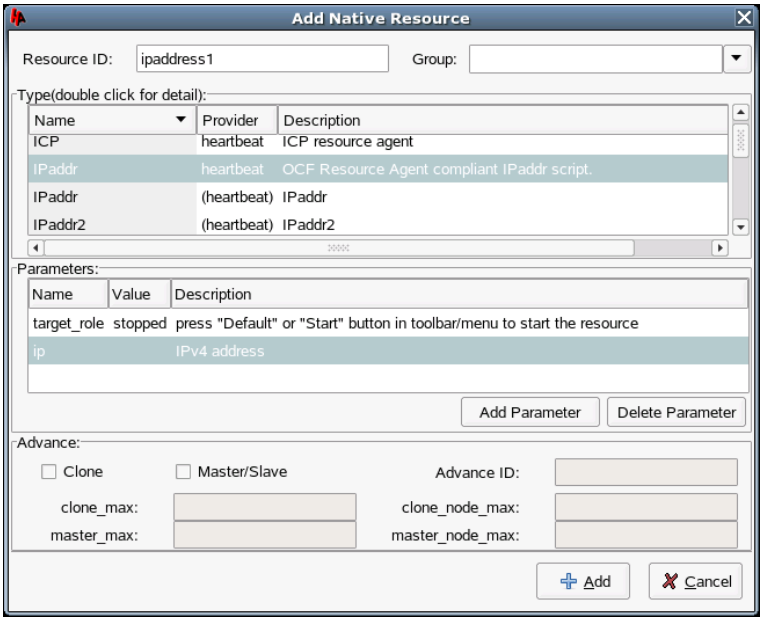
5 Select the resource *Type* from the list. To display a brief description of that resource type, double-click any of the resource types in the list.

6 Conditional: After selecting a resource type, a line for that resource type might be added to the *Parameters* section of the screen. If a line is added, click the line once and then click the line again under the *Value* heading to open a field where you can add the needed value.

6a Add the needed value for the resource.

For example, if you selected an IP address as the resource type above, enter the IP address for the resource.

6b Click *Add Parameter*, specify the parameter *Name* and *Value*, then click *OK*.



7 Click *Add* at the bottom of the screen to add the resource to the cluster.

4.3 Configuring Resource Constraints

Resource constraints let you specify which cluster nodes resources will run on, what order resources will load, and what other resources a specific resource is dependent on. For information on configuring resource constraints, see Constraints [http://linux-ha.org/ClusterInformationBase/SimpleExamples] on the High Availability Linux Web site.

4.4 Specifying Resource Failover Nodes

Cluster resources will automatically fail over to another node in the cluster in the event of a software or hardware failure. If you have more than two nodes in your cluster, the node a particular resource fails over to is chosen by the Heartbeat software. If you want to choose which node a resource will fail over to, you must configure a location constraint for that resource.

1. Start the HA Management Client and log in to the cluster as described in Section 4.1, “Graphical HA Management Client” (page 29).
2. Select the desired resource or group, select *Resources* > *Add New Item*, or click the + button.
3. Choose *Location* as the resource item type, then click *OK*.
4. Specify the *ID* of the resource constraint. The resource name for the resource you selected should already be displayed in the *Resource* field.
5. Click *OK* to add the constraint. The score will automatically be set to `0` by default to avoid unwanted effects caused by an incomplete rule.

   Score indicates the value you are assigning to this resource constraint. Constraints with higher scores are applied before those with lower scores. By creating additional location constraints with different scores for a given resource, you can specify an order for the nodes that a resource will fail over to.

6 If you need to modify the score, select the added constraint in the left pane of the main window and enter a new *Score* value in the *Attributes* section in the right pane.

7 On the left side of the page, select the resource constraint you just created and click *Add Expression*.

8 Select *#uname* as the *Attribute*, *eq* as the *Operation*, and the server name of the failover node as the *Value*.

9 Click *OK,* then click *Apply* to save your changes.

10 Conditional: If you want to specify additional failover nodes for a resource, create additional location constraints by performing Step 1 (page 33) through Step 9 (page 34) for each constraint.

4.5 Specifying Resource Failback Nodes (Resource Stickiness)

A resource will automatically fail back to its original node when that node is back online and in the cluster. If you want to prevent a resource from failing back to the node it was running on prior to a failover, or if you want to specify a different node for the resource to fail back to, you must change its resource stickiness value. You can specify resource stickiness when you are creating a resource, or after.

To specify resource stickiness for a resource after it has been created:

1 Start the HA Management Client and log in to the cluster as described in Section 4.1, “Graphical HA Management Client” (page 29).

2 Select the desired resource or group, click the *Attributes* tab, then click *Add Attribute*.

3 In the *Name* field, type `resource_stickiness`.

4 In the *Value* field, specify a value between `-100,000` and `100,000`.

Consider the following when specifying a resource stickiness value:

Value is `0`:

This is the default. The resource will be placed optimally in the system. This may mean that it is moved when a "better" or less loaded node becomes available. This option is almost equivalent to automatic failback, except that the resource may be moved to a node that is not the one it was previously active on.

Value is greater than `0`:

The resource will prefer to remain in its current location, but may be moved if a more suitable node is available. Higher values indicate a stronger preference for a resource to stay where it is.

Value is less than `0`:

The resource prefers to move away from its current location. Higher absolute values indicate a stronger preference for a resource to be moved.

Value is `100,000`:

The resource will always remain in its current location unless forced off because the node is no longer eligible to run the resource (node shutdown, node standby, or configuration change). This option is almost equivalent to completely disabling automatic failback, except that the resource may be moved to other nodes than the one it was previously active on.

Value is `-100,000`:

The resource will always move away from its current location.

See also the `crm_failcount` command: crm_failcount(8) (page 81).

4.6 Configuring Resource Monitoring

Although Heartbeat can detect a node failure, it also has the ability to detect when an individual resource on a node has failed. If you want Heartbeat to ensure that a resource is running, you must configure resource monitoring for that resource. Resource monitoring consists of specifying a timeout and/or start delay value, and an interval. The interval tells Heartbeat how often it should check the resource status.

To configure resource monitoring:

1. Start the HA Management Client and log in to the cluster as described in Section 4.1, "Graphical HA Management Client" (page 29).

2 Select the resource, click the *Operations* tab, then click *Add Operation*.

3 Select *Monitor* as the operation name.

4 Add the desired values in the *Interval*, *Timeout*, and *Start Delay* fields, and a description if desired.

5 Click *OK*, then click *Apply* to start the monitoring operation.

If you do not configure resource monitoring, resource failures after a successful start will not be communicated, and the cluster will always show the resource as healthy.

If the resource monitor detects a failure, the following actions will take place:

- Log file messages will be generated
- The failure will be reflected in the hb_gui and crm_mon tools, and in the CIB status section
- The cluster will initiate noticeable recovery actions which may include stopping the resource to repair the failed state and restarting the resource locally or on another node. The resource also may not be restarted at all, depending on the configuration and state of the cluster.

4.7 Starting a New Cluster Resource

Some cluster resources do not start automatically after being created. To start a new cluster resource after creating it, in the HA Management Client, select the resource in the left pane, then click *Resource* > *Start*.

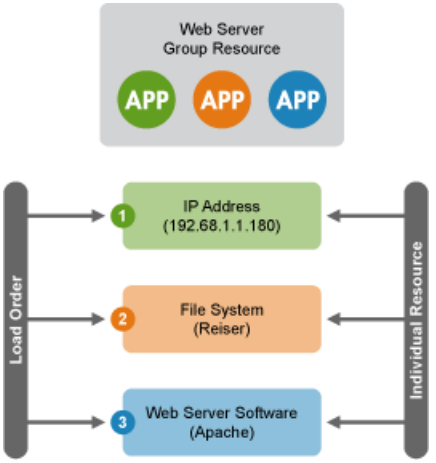
4.8 Removing a Cluster Resource

You can remove a cluster resource using the HA Management Client. To do this, select the resource in the left pane, and click *Resource* > *Delete*.

4.9 Configuring a Heartbeat Cluster Resource Group

Some cluster resources are dependent on other components or resources, and require that each component or resource are started in a specific order and run together on the same server. An example of this is a Web server that requires an IP address and a file system. In this case each component is a separate cluster resource that is combined into a cluster resource group. The resource group would then run on a server or servers, and in case of a software or hardware malfunction, fail over to another server in the cluster the same as an individual cluster resource.

Figure 4.1 *Group Resource*



To configure a resource group using the above example:

1. Start the HA Management Client and log in to the cluster as described in Section 4.1, “Graphical HA Management Client” (page 29).
2. Select *Resources* > *Add New Item*, or click the + button.
3. Choose *Group* as the resource item type, then click *OK*.
4. Specify a Group name (ID), leave the default of `True` for both the *Ordered* and *Collocated* fields, then click *OK*.

The *Ordered* value specifies, if the resources in the group will load in order you specified. The *Collocated* value specifies whether the resources in the group will run on the same server.

5 Specify a resource name (ID) for the IP address resource portion of the group.

6 In the *Type* section of the page, scroll down the list and select *IPaddr2 (OCF Resource Agent)* as the resource type.

7 In the *Parameters* section of the page, find the line that was added for the IP address resource, click the line once, then click the line again under the *Value* heading to open a text field.

8 Add the IP address for the IP address cluster resource.

9 Click the *Add Parameter* button, select *nic* as the name and enter `eth0` as the value, then click *OK*.

The name and value are dependent on your hardware configuration and what you chose in Step 3 (page 31).

10 Click the *Add* button at the bottom of the page to add the resource to the group.

4.9.1 Adding a File System to the Resource Group

1 Using the HA Management Client, select *Resources,* and then select *Add New Item*, or click the + button.

2 Choose *Native* as the resource item type, then click *OK*.

3 Specify a resource name (ID) for the file system resource, and in the *Belong to Group* field, select the group you created above.

4 In the *Type* section of the page, scroll down the list and select *Filesystem* as the resource type.

5 In the *Parameters* section of the page, find the line that was added for the file system resource, click the line once, then click the line again under the *Value* heading to open a text field.

6 Add the name of the file system. For example, `Reiser`.

7 Click the *Add Parameter* button and select *Device* as the name. For the value, specify the device where the file system is located, then click *OK*.

The name and value are dependent on your hardware configuration. For example, you could specify `/dev/sdc` as the value if the file system is on that device.

4.9.2 Adding the Web Server to the Resource Group

1 Using the HA Management Client, select *Resources,* and then select *Add New Item*, or click the + button.

2 Choose *Native* as the resource item type, then click *OK*.

3 Specify a resource name (ID) for the Web Server resource, and in the *Belong to group* field, select the group you created above.

4 In the *Type* section of the page, scroll down the list and select Apache as the resource type.

5 In the *Parameters* section of the page, find the line that was added for the Apache Web server resource, accept or change the path to the Apache Web server, then click *OK*.

NOTE

If you want to enable resource monitoring for a group resource, you must configure monitoring separately for each resource in the group that you want monitored.

4.10 Configuring a Heartbeat Clone Resource

You may want certain resources to run simultaneously on multiple nodes in your cluster. To do this you must configure a resource as a clone. Examples of resources that might be configured as clones include STONITH and cluster file systems like OCFS2. You can clone any resource provided it is supported by the resource's ResourceAgent. Clone resources may even be configured differently depending on which nodes they are hosted.

There are three types of resource clones:

- **Anonymous Clones:** These are the simplest type of clones. They behave identically anywhere they are running.
- **Globally Unique Clones:** These resources are distinct entities. An instance of the clone running on one node is not equivalent to another instance on another node; nor would any two instances on the same node be equivalent.
- **Stateful Clones:** Active instances of these resources are divided into two states, active and passive. These are also sometimes referred to as primary and secondary, or master and slave. Stateful clones can be either anonymous or globally unique.

1. To configure a resource as a clone, follow Step 1 (page 31) through Step 6 (page 32) in Section 4.2, "Creating Cluster Resources" (page 31).
2. Select the *Clone* check box. This will allow the resource to simultaneously run on multiple cluster nodes.
3. In the *clone_node_max* field, enter the number of instances of the resource that will run on a given node.
4. In the *clone_max* field, enter the number of nodes in the cluster that will run the resource.
5. Click *Add* to add the clone resource to the cluster.

6 Select the resource in the left pane and click *Resource* > *Start* to start the resource. Starting the clone resource will cause it to start on the nodes that have been specified with its resource location constraints.

After creating the resource, you must create location constraints for the resource. The location constraints determine which nodes the resource can run on. See Constraints [http://www.linux-ha.org/ClusterInformationBase/SimpleExamples] on the High Availability Linux Web site.

4.11 Migrating a Cluster Resource

Although resources are configured to automatically fail over (or migrate) to other nodes in the cluster in the even of a hardware or software failure, you can also manually migrate a resource to another node in the cluster using either the HA Management Client or the command line.

1 At the command line, use the following command:

```
crm_resource -M -r resource_name -H hostname -f
```

For example, to migrate a resource named `ipaddress1` to a cluster node named `node2`, you would need to enter:

```
crm_resource -M -r ipaddress1 -H node2 -f
```

To get a list of parameters and switches that can be used with the `crm_resource` command, enter `crm_resource` without any options.

2 To use the HA Management Client for migration, start the HA Management Client and log in to the cluster as described in Section 4.1, “Graphical HA Management Client” (page 29).

3 Select the resource you want to migrate in the left pane of the main window and select *Resources* > *Migrate Resource*.

4 In the new window, select `ipaddress1` as *Resource* to migrate and select `node1` from the *To Node* drop-down list.

Manual Configuration of a Cluster

5

Manual configuration of a Heartbeat cluster is often the most effective way of creating a reliable cluster that meets specific needs. Because of the extensive configurability of Heartbeat and the range of needs it can meet, it is not possible to document every possible scenario. To introduce several concepts of the Heartbeat configuration and demonstrate basic procedures, consider a real world example of an NFS file server. The goal is to create an NFS server that can be built with very low-cost parts and is as redundant as possible. For this, set up the following cluster:

- Two machines that have redundant hardware
- Data is mirrored on the disks of those machines with drbd
- Only one machine at a time accesses and exports the data
- Assign a special IP address to the computer for exporting the file system

Before starting with the cluster configuration, set up two nodes as described in Chapter 2, *Installation and Setup* (page 13). In addition to the system installation, both should have a data partition of the same size to setup drbd.

The configuration splits into two main parts. First, all the resources must be configured. After this, create a set of `constraints` that define the starting rules for those resources.

All the configuration data is written in XML. For convenience, the example relies on snippets that may be loaded into the cluster configuration individually.

5.1 Configuration Basics

The cluster is divided into two main sections, configuration and status. The status section contains the history of each resource on each node and based on this data, the cluster can construct the complete current state of the cluster. The authoritative source for the status section is the *local resource manager* (lrmd) process on each cluster node. The cluster will occasionally repopulate the entire section. For this reason it is never written to disk and administrators are advised against modifying it in any way.

The configuration section contains the more traditional information like cluster options, lists of resources and indications of where they should be placed. It is the primary focus of this document and is divided into four parts:

- Configuration options (called crm_config)
- Nodes
- Resources
- Resource relationships (called *constraints*)

Example 5.1 *Structure of an Empty Configuration*

```
<cib generated="true" admin_epoch="0" epoch="0" num_updates="0"
have_quorum="false">
  <configuration>
    <crm_config/>
    <nodes/>
    <resources/>
    <constraints/>
  </configuration>
  <status/>
</cib>
```

5.1.1 The Current State of the Cluster

Before you start to configure a cluster, it is worth explaining how to view the finished product. For this purpose use the `crm_mon` utility that will display the current state of an active cluster. It can show the cluster status by node or by resource and can be used in either single-shot or dynamically-updating mode. Using this tool, you can examine the state of the cluster for irregularities and see how it responds when you cause or simulate failures.

Details on all the available options can be obtained using the `crm_mon --help` command.

5.1.2 Updating the Configuration

There is a basic warning for updating the cluster configuration:

WARNING: Rules For Updating the Configuration

Never edit the `cib.xml` file manually, otherwise the cluster will refuse to use the configuration. Instead, always use the `cibadmin` tool to change your configuration.

To modify your cluster configuration, use the `cibadmin` command which talks to a running cluster. With `cibadmin`, you can query, add, remove, update or replace any part of the configuration. All changes take effect immediately and there is no need to perform a reload-like operation.

The simplest way of using `cibadmin` is a three-step procedure:

1. Save the current configuration to a temporary file:

   ```
   cibadmin --cib_query > /tmp/tmp.xml
   ```

2. Edit the temporary file with your favorite text or XML editor.

   Some of the better XML editors are able to use the DTD (document type definition) to make sure that any changes you make are valid. The DTD describing the configuration can be found in `/usr/lib/heartbeat/crm.dtd` on your systems.

3. Upload the revised configuration:

   ```
   cibadmin --cib_replace --xml-file /tmp/tmp.xml
   ```

If you only want to modify the `resources` section, do the following to avoid modifying any other part of the configuration:

```
cibadmin --cib_query --obj_type resources > /tmp/tmp.xml
vi /tmp/tmp.xml
cibadmin --cib_replace --obj_type resources --xml-file /tmp/tmp.xml
```

5.1.3 Quickly Deleting Part of the Configuration

Sometimes it is necessary to delete an object quickly. This can be done in three easy steps:

1. Identify the object you wish to delete, for example:

```
cibadmin -Q | grep stonith
 <nvpair id="cib-bootstrap-options-stonith-action" name="stonith-action"
value="reboot"/>
 <nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled"
value="1"/>
 <primitive id="child_DoFencing" class="stonith" type="external/vmware">
 <lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
 <lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
 <lrm_resource id="child_DoFencing:1" type="external/vmware" class="stonith">
 <lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
 <lrm_resource id="child_DoFencing:2" type="external/vmware" class="stonith">
 <lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
 <lrm_resource id="child_DoFencing:3" type="external/vmware" class="stonith">
```

2. Identify the resource's tag name and id (in this case `primitive` and `child_DoFencing`.

3. Execute `cibadmin`:

```
cibadmin --cib_delete --crm_xml '<primitive id="child_DoFencing"/>'
```

5.1.4 Updating the Configuration Without Using XML

Some common tasks can also be performed with one of the higher level tools that avoid the need to read or edit XML. Run the following command to enable `STONITH`, for example:

```
crm_attribute --attr-name stonith-enabled --attr-value true
```

Or to see if `somenode` is allowed to run resources, there is:

```
crm_standby --get-value --node-uname somenode
```

Or to find the current location of `my-test-rsc` one can use:

```
crm_resource --locate --resource my-test-rsc
```

5.1.5 Testing Your Configuration

It is not necessary to modify a real cluster in order to test the effect of the configuration changes. Do the following to test your modifications:

1. Save the current configuration to a temporary file:

   ```
   cibadmin --cib_query > /tmp/tmp.xml
   ```

2. Edit the temporary file with your favorite text or XML editor.

3. Simulate the effect of the changes with `ptest`:

   ```
   ptest -VVVVV --xml-file /tmp/tmp.xml --save-graph tmp.graph
   --save-dotfile tmp.dot
   ```

The tool uses the same library as the live cluster to show the impact it would have done. Its output, in addition to a significant amount of logging, is stored in two files, `tmp.graph` and `tmp.dot`. Both files are representations of the same thing—the cluster's response to your changes. In the graph file the complete transition is stored, containing a list of all actions, their parameters and their prerequisites. The transition graph is not very easy to read. Therefore, the tool also generates a `Graphviz` dot-file representing the same information.

5.2 Configuring Resources

There are three types of RAs (Resource Agents) available with Heartbeat. First, there are legacy Heartbeat 1 scripts. Heartbeat can make use of LSB initialization scripts. Finally, Heartbeat has its own set of OCF (Open Cluster Framework) agents. This documentation concentrates on LSB scripts and OCF agents.

5.2.1 LSB Initialization Scripts

All LSB scripts are commonly found in the directory `/etc/init.d`. They must have several actions implemented, which are at least `start`, `stop`, `restart`, `reload`, `force-reload`, and `status` as explained in http://www.linux-foundation.org/spec/refspecs/LSB_1.3.0/gLSB/gLSB/iniscrptact.html.

The configuration of those services is not standardized. If you intend to use an LSB script with Heartbeat, make sure that you understand how the respective script is configured. Often you can find some documentation to this in the documentation of the respective package in `/usr/share/doc/packages/<package_name>`.

When used by Heartbeat, the service should not be touched by other means. This means that it should not be started or stopped on boot, reboot, or manually. However, if you want to check if the service is configured properly, start it manually, but make sure that it is stopped again before Heartbeat takes over.

Before using an LSB resource, make sure that the configuration of this resource is present and identical on all cluster nodes. The configuration is not managed by Heartbeat. You must take care of that yourself.

5.2.2 OCF Resource Agents

All OCF agents are located in `/usr/lib/ocf/resource.d/heartbeat/`. These are small programs that have a functionality similar to that of LSB scripts. However, the configuration is always done with environment variables. All OCF Resource Agents are required to have at least the actions `start`, `stop`, `status`, `monitor`, and `meta-data`. The `meta-data` action retrieves information about how to configure the agent. For example, if you want to know more about the `IPaddr` agent, use the command:

```
/usr/lib/ocf/resource.d/heartbeat/IPaddr meta-data
```

The output is lengthy information in a simple XML format. You can validate the output with the `ra-api-1.dtd` DTD. Basically this XML format has three sections—first several common descriptions, second all the available parameters, and last the available actions for this agent.

A typical parameter of a OCF RA as shown with the `meta-data` command looks like this:

```
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="apache"> ❶
  <!-- Some elements omitted -->
  <parameter name="ip" unique="1" required="1">❷
    <longdesc lang="en">❸
The IPv4 address to be configured in dotted quad notation, for example
"192.168.1.1".
    </longdesc>
    <shortdesc lang="en">IPv4 address</shortdesc>
    <content type="string" default="" />❹
  </parameter>
</resource-agent>
```

This is part of the `IPaddr` RA. The information about how to configure the parameter of this RA can be read as follows:

❶ Root element for each output.

❷ The name of the `nvpair` to configure is `ip`. This RA attribute is mandatory for the configuration.

❸ The description of the parameter is available in a long and a short description tag.

❹ The content of the value of this parameter is a string. There is no default value available for this resource.

Find a configuration example for this RA at Chapter 3, *Setting Up a Simple Resource* (page 25).

5.2.3 Example Configuration for an NFS Server

To set up the NFS server, three resources are needed: a file system resource, a drbd resource, and a group of an NFS server and an IP address. You can write each of the resource configurations to a separate file then load them to the cluster with `cibadmin -C -o resources -x` *`resource_configuration_file`*.

Setting Up a File System Resource

The `filesystem` resource is configured as an OCF primitive resource. It has the task to mount and unmount a device to a directory on start and stop requests. In this case, the device is `/dev/drbd0` and the directory to use as mount point is `/srv/failover`. The file system used is `reiserfs`.

The configuration for this resource looks like the following:

```
<primitive id="filesystem_resource" class="ocf" provider="heartbeat"
type="Filesystem">
  <instance_attributes id="ia-filesystem_1">
    <attributes>
      <nvpair id="filesystem-nv-1" name="device" value="/dev/drbd0"/>
      <nvpair id="filesystem-nv-2" name="directory" value="/srv/failover"/>
      <nvpair id="filesystem-nv-3" name="fstype" value="reiserfs"/>
    </attributes>
  </instance_attributes>
</primitive>
```

Configuring drbd

Before starting with the drbd Heartbeat configuration, set up a `drbd` device manually. Basically this is configuring drbd in `/etc/drbd.conf` and letting it synchronize. The exact procedure for configuring drbd is described in the *Storage Administration Guide*. For now, assume that you configured a resource `r0` that may be accessed at the device `/dev/drbd0` on both of your cluster nodes.

The `drbd` resource is an OCF master slave resource. This can be found in the description of the metadata of the drbd RA. However, more important is that there are the actions `promote` and `demote` in the `actions` section of the metadata. These are mandatory for master slave resources and commonly not available to other resources.

For Heartbeat, master slave resources may have multiple masters on different nodes. It is even possible to have a master and slave on the same node. Therefore, configure this resource in a way that there is exactly one master and one slave, each running on different nodes. Do this with the meta attributes of the `master_slave` resource. Master slave resources are a special kind of clone resources in Heartbeat. Every master and every slave counts as a clone.

```
<master_slave id="drbd_resource" ordered="false">❶
  <meta_attributes>
    <attributes>
      <nvpair id="drbd-nv-1" name="clone_max" value="2"/> ❷
      <nvpair id="drbd-nv-2" name="clone_node_max" value="1"/>❸
      <nvpair id="drbd-nv-3" name="master_max" value="1"/>❹
      <nvpair id="drbd-nv-4" name="master_node_max" value="1"/>❺
      <nvpair id="drbd-nv-5" name="notify" value="yes"/>❻
    </attributes>
  </meta_attributes>
  <primitive id="drbd_r0" class="ocf" provider="heartbeat" type="drbd">❼
    <instance_attributes id="ia-drbd_1">
      <attributes>
        <nvpair id="drbd-nv-5" name="drbd_resource" value="r0"/>❽
      </attributes>
    </instance_attributes>
  </primitive>
</master_slave>
```

❶ The master element of this resource is `master_slave`. The complete resource is later accessed with the ID `drbd_resource`.

❷ `clone_max` defines how many masters and slaves may be present in the cluster.

❸ `clone_node_max` is the maximum number of clones (masters or slaves) that are allowed to run on a single. node.

❹ `master_max` sets how many masters may be available in the cluster.

❺ `master_node_max` is similar to `clone_node_max` and defines how many master instances may run on a single node.

❻ `notify` is used to inform the cluster before and after a clone of the `master_slave` resource is stopped or started. This is used to reconfigure one of the clones to be a master of this resource.

❼ The actually working RA inside this `master slave` resource is the drbd primitive.

❽ The most important parameter this resource needs to know about is the name of the drbd resource to handle.

NFS Server and IP Address

To make the NFS server always available at the same IP address, use an additional IP address as well as the ones the machines use for their normal operation. This IP address is then assigned to the active NFS server in addition to the system's IP address.

The NFS server and the IP address of the NFS server should always be active on the same machine. In this case, the start sequence is not very important. They may even be started at the same time. These are the typical requirements for a group resource.

Before starting the Heartbeat RA configuration, configure the NFS server with YaST. Do not let the system start the NFS server. Just set up the configuration file. If you want to do that manually, see the manual page exports(5) (`man 5 exports`. The configuration file is `/etc/exports`. The NFS server is configured as an LSB resource.

Configure the IP address completely with the Heartbeat RA configuration. No additional modification is necessary in the system. The IP address RA is an OCF RA.

```
<group id="nfs_group">❶
  <primitive id="nfs_resource" class="lsb" type="nfsserver"/>❷
  <primitive id="ip_resource" class="ocf" provider="heartbeat"
    type="IPaddr">❸
    <instance_attributes id="ia-ipaddr_1">
      <attributes>
        <nvpair id="ipaddr-nv-1" name="ip" value="10.10.0.1"/>❹
      </attributes>
    </instance_attributes>
  </primitive>
</group>
```

❶ In a group resource, there may be several other resources. It must have an ID set.

❷ The `nfsserver` is simple. It is just the LSB script for the NFS server. The service itself must be configured in the system.

❸ The `IPaddr` OCF RA does not need any configuration in the system. It is just configured with the following `instance_attributes`.

❹ There is only one mandatory instance attribute in the `IPaddr` RA. More possible configuration options are found in the metadata of the RA.

5.3 Configuring Constraints

Having all the resources configured is only part of the job. Even if the clusters knows all needed resources, it might still not be able to handle them correctly. For example, it would be quite useless to try to mount the file system on the slave node of drbd (in fact, this would fail with drbd). To inform the cluster about these things, define constraints.

In Heartbeat, there are three different kinds of constraints available:

- Locational constraints that define on which nodes a resource may be run (`rsc_location`).
- Colocational constraints that tell the cluster which resources may or may not run together on a node (`rsc_colocation`).
- Ordering constraints to define the sequence of actions (`rsc_order`).

5.3.1 Locational Constraints

This type of constraint may be added multiple times for each resource. All `rsc_location` constraints are evaluated for a given resource. A simple example that increases the probability to run a resource with the ID `filesystem_1` on the node with the name `earth` to 100 would be the following:

```
<rsc_location id="filesystem_1_location" rsc="filesystem_1">❶
   <rule id="pref_filesystem_1" score="100">❷
      <expression attribute="#uname" operation="eq" value="earth"/>❸
   </rule>
</rsc_location>
```

❶ To take effect, the `rsc_location` tag must define an `rsc` attribute. The content of this attribute must be the ID of a resource of the cluster.

❷ The `score` attribute is set by the `rule` tag depending on the following `expression`, and is used as a priority to run a resource on a node. The scores are calculated by rules (see http://wiki.linux-ha.org/ScoreCalculation for details.)

❸ Whether a `rule` really is activated, changing the `score`, depends on the evaluation of an `expression`. Several different operations are defined and the special attributes `#uname` and `#id` may be used in the comparison.

It is also possible to use another rule or a `date_expression`. For more information, refer to `crm.dtd`, which is located at `/usr/lib/heartbeat/crm.dtd`.

5.3.2 Colocational Constraints

The `rsc_colocation` constraint is used to define what resources should run on the same or on different hosts. It is not possible to give a score other than `INFINITY` or `-INFINITY`, defining resources to run together always or never to run together. For example, to run the two resources with the IDs `filesystem_resource` and `nfs_group` always on the same host, use the following constraint:

```
<rsc_colocation
  id="nfs_on_filesystem"
  to="filesystem_resource"
  from="nfs_group"
  score="INFINITY"/>
```

For a master slave configuration, it is necessary to know if the current node is a master in addition to running the resource locally. This can be checked with an additional `to_role` or `from_role` attribute.

5.3.3 Ordering Constraints

Sometimes it is necessary to provide an order in which services must start. For example, you cannot mount a file system before the device is available to a system. Ordering constraints can be used to start or stop a service right before or after a different resource meets a special condition, such as being started, stopped, or promoted to master. An ordering constraint looks like the following:

```
<rsc_order id="nfs_after_filesystem" from="group_nfs" action="start"
      to="filesystem_resource" to_action="start" type="after"/>
```

With `type="after"`, the `action` of the `from` resource is done after the action of the `to` resource.

5.3.4 Constraints for the Example Configuration

The example used for this chapter is quite useless without additional constraints. It is essential that all resources run on the same machine as the master of the drbd resource. Another thing that is critical is that the drbd resource must be master before any other

resource starts. Trying to mount the drbd device when drbd is not master simply fails. The constraints that must be fulfilled look like the following:

- The file system must always be on the same node as the master of the drbd resource.

```
<rsc_colocation id="filesystem_on_master" to="drbd_resource"
       to_role="master" from="filesystem_resource" score="INFINITY"/>
```

- The file system must be mounted on a node after the drbd resource is promoted to master on this node.

```
<rsc_order id="drbd_first" from="filesystem_resource" action="start"
       to="drbd_resource" to_action="promote" type="after"/>
```

- The NFS server as well as the IP address start after the file system is mounted.

```
<rsc_order id="nfs_second" from="nfs_group" action="start"
       to="filesystem_resource" to_action="start" type="after"/>
```

- The NFS server as well as the IP address must be on the same node as the file system.

```
<rsc_colocation id="nfs_on_drbd" to="filesystem_resource"
       from="nfs_group" score="INFINITY"/>
```

- In addition to that, issue constraint that prevents the NFS server from running on a node where drbd is running in slave mode.

```
<rsc_colocation id="nfs_on_slave" to="drbd_resource"
       to_role="slave" from="nfs_group" score="-INFINITY"/>
```

5.4 Configuring CRM Options

The CRM options define the global behavior of a cluster. In principle, the default values should be acceptable for many environments, but if you want to use special services, like STONITH devices, you must inform the cluster about this. All options of `crm_config` are made with `nvpair` and are added to `cib.xml`. For example, to change the `cluster-delay` from its default value of `60s` to `120s`, use the following configuration:

```
<cluster_property_set>
  <attributes>
     <nvpair id="1" name="cluster-delay" value="120s"/>
  </attributes>
</cluster_property_set>
```

Write this information to a file and load it to the cluster with the command `cibadmin -C -o crm_config -x` *`filename`*. The following is an overview of all available configuration options:

`cluster-delay` (interval, default=60s)
: This option used to be known as `transition_idle_timeout`. If no activity is recorded in this time, the transition is deemed failed as are all sent actions that have not yet been confirmed complete. If any operation initiated has an explicit higher time-out, the higher value applies.

`symmetric_cluster` (boolean, default=TRUE)
: If true, resources are permitted to run anywhere by default. Otherwise, explicit constraints must be created to specify where they can run.

`stonith_enabled` (boolean, default=FALSE)
: If true, failed nodes are fenced.

`no_quorum_policy` (enum, default=stop)

`ignore`
: Pretend to have quorum.

`freeze`
: Do not start any resources not currently in the partition. Resources in the partition may be moved to another node within the partition. Fencing is disabled.

`stop`
: Stop all running resources in the partition. Fencing is disabled.

`default_resource_stickiness` (integer, default=0)
: Select whether resources should prefer to run on the existing node or be moved to a “better” one?

`0`
: Resources are placed optimally in the system. This may mean they are moved when a "better" or less-loaded node becomes available. This option is almost equivalent to `auto_failback` on except that the resource may be moved to nodes other than the one on which it was previously active.

`value > 0`
: Resources prefer to remain in their current location but may be moved if a more suitable node is available. Higher values indicate a stronger preference for resources to stay where they are.

`value < 0`
: Resources prefer to move away from their current location. Higher absolute values indicate a stronger preference for resources to be moved.

`INFINITY`
: Resources always remain in their current locations until forced off because the node is no longer eligible to run the resource (node shutdown, node standby, or configuration change). This option is almost equivalent to `auto_failback` off except that the resource may be moved to other nodes than the one on which it was previously active.

`-INFINITY`
: Resources always move away from their current location.

`is_managed_default` (boolean, default=TRUE)
: Unless the resource's definition says otherwise:

`TRUE`
: Resources are started, stopped, monitored, and moved as necessary.

`FALSE`
: Resources are not started if stopped, stopped if started, or have any recurring actions scheduled.

`stop_orphan_resources` (boolean, default=TRUE)
: If a resource is found for which there is no definition:

`TRUE`
: Stop the resource.

`FALSE`
: Ignore the resource.

This mostly affects the CRM's behavior when a resource is deleted by an administrator without it first being stopped.

`stop_orphan_actions` (boolean, default=TRUE)
: If a recurring action is found for which there is no definition:

`TRUE`
: Stop the action.

`FALSE`
: Ignore the action.

All available options to the `crm_config` are summarized in Policy Engine(7) (page 59).

Policy Engine (7)

Policy Engine — Policy Engine Options

Synopsis

[no-quorum-policy=enum] [symmetric-cluster=boolean]
[stonith-enabled=boolean] [stonith-action=enum]
[default-resource-stickiness=integer]
[default-resource-failure-stickiness=integer]
[is-managed-default=boolean] [cluster-delay=time]
[default-action-timeout=time] [stop-orphan-resources=boolean]
[stop-orphan-actions=boolean] [pe-error-series-max=integer]
[pe-warn-series-max=integer] [pe-input-series-max=integer]
[startup-fencing=boolean]

Description

This is a fake resource that details the options that can be configured for the Policy Engine.

Supported Parameters

no-quorum-policy=What to do when the cluster does not have quorum
What to do when the cluster does not have quorum Allowed values: stop, freeze, ignore

symmetric-cluster=All resources can run anywhere by default
All resources can run anywhere by default

stonith-enabled=Failed nodes are STONITH'd
Failed nodes are STONITH'd

stonith-action=Action to send to STONITH device
Action to send to STONITH device Allowed values: reboot, poweroff

`default-resource-stickiness=`

`default-resource-failure-stickiness=`

`is-managed-default`=Should the cluster start/stop resources as required
Should the cluster start/stop resources as required

`cluster-delay`=Round trip delay over the network (excluding action execution)
The "correct" value will depend on the speed and load of your network and cluster nodes.

`default-action-timeout`=How long to wait for actions to complete
How long to wait for actions to complete

`stop-orphan-resources`=Should deleted resources be stopped
Should deleted resources be stopped

`stop-orphan-actions`=Should deleted actions be cancelled
Should deleted actions be cancelled

`pe-error-series-max`=The number of PE inputs resulting in ERRORs to save
Zero to disable, -1 to store unlimited.

`pe-warn-series-max`=The number of PE inputs resulting in WARNINGs to save
Zero to disable, -1 to store unlimited.

`pe-input-series-max`=The number of other PE inputs to save
Zero to disable, -1 to store unlimited.

`startup-fencing`=STONITH unseen nodes
Advanced Use Only! Not using the default is very unsafe!

5.5 For More Information

`http://linux-ha.org`
Homepage of High Availability Linux

6 Managing a Cluster

Heartbeat ships with a comprehensive set of tools that assist you in managing your cluster from the command line. This chapter introduces the tools needed for managing the cluster configuration in the CIB and the cluster resources. Other command line tools for managing resource agents or tools used for debugging and troubleshooting your setup are covered in Chapter 7, *Creating Resources* (page 103) and Chapter 8, *Troubleshooting* (page 107).

The following list presents several tasks related to cluster management and briefly introduces the tools to use to accomplish these tasks:

Monitoring the Cluster's Status

The `crm_mon` command allows you to monitor your cluster's status and configuration. Its output includes the number of nodes, uname, uuid, status, the resources configured in your cluster, and the current status of each. The output of `crm_mon` can be displayed at the console or printed into an HTML file. When provided with a cluster configuration file without the status section, `crm_mon` creates an overview of nodes and resources as specified in the file. See crm_mon(8) (page 86) for a detailed introduction to this tool's usage and command syntax.

Managing the CIB

The `cibadmin` command is the low-level administrative command for manipulating the Heartbeat CIB. It can be used to dump all or part of the CIB, update all or part of it, modify all or part of it, delete the entire CIB, or perform miscellaneous CIB administrative operations. See cibadmin(8) (page 64) for a detailed introduction to this tool's usage and command syntax.

Managing Configuration Changes

The `crm_diff` command assists you in creating and applying XML patches. This can be useful for visualizing the changes between two versions of the cluster configuration or saving changes so they can be applied at a later time using cibadmin(8) (page 64). See crm_diff(8) (page 78) for a detailed introduction to this tool's usage and command syntax.

Manipulating CIB Attributes

The `crm_attribute` command lets you query and manipulate node attributes and cluster configuration options that are used in the CIB. See crm_attribute(8) (page 74) for a detailed introduction to this tool's usage and command syntax.

Validating the Cluster Configuration

The `crm_verify` command checks the configuration database (CIB) for consistency and other problems. It can check a file containing the configuration or connect to a running cluster. It reports two classes of problems. Errors must be fixed before Heartbeat can work properly while warning resolution is up to the administrator. `crm_verify` assists in creating new or modified configurations. You can take a local copy of a CIB in the running cluster, edit it, validate it using `crm_verify`, then put the new configuration into effect using `cibadmin`. See crm_verify(8) (page 100) for a detailed introduction to this tool's usage and command syntax.

Managing Resource Configurations

The `crm_resource` command performs various resource-related actions on the cluster. It lets you modify the definition of configured resources, start and stop resources, or delete and migrate resources between nodes. See crm_resource(8) (page 89) for a detailed introduction to this tool's usage and command syntax.

Managing Resource Fail Counts

The `crm_failcount` command queries the number of failures per resource on a given node. This tool can also be used to reset the failcount, allowing the resource to again run on nodes where it had failed too often. See crm_failcount(8) (page 81) for a detailed introduction to this tool's usage and command syntax.

Generate and Retrieve Node UUIDs

UUIDs are used to identify cluster nodes to ensure that they can always be uniquely identified. The command `crm_uuid` displays the UUID of the node on which it is run. In very rare circumstances, it may be necessary to set a node's UUID to a known value. This can also be achieved with `crm_uuid`, but you should use

this command with extreme caution. For more information, refer to crm_uuid(8) (page 98).

Managing a Node's Standby Status

The `crm_standby` command can manipulate a node's standby attribute. Any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Standby mode can be useful for performing maintenance tasks, such as kernel updates. Remove the standby attribute from the node as it should become a fully active member of the cluster again. See crm_standby(8) (page 95) for a detailed introduction to this tool's usage and command syntax.

cibadmin (8)

cibadmin — read, modify, or administer Heartbeat Cluster Information Base

Synopsis

```
cibadmin (--cib_query|-Q) -[Vrwlsmfbp] [-i xml-object-id|-o
xml-object-type] [-t t-flag-whatever] [-h hostname]

cibadmin (--cib_create|-C) -[Vrwlsmfbp] [-X xml-string]
[-x xml- filename] [-t t-flag-whatever] [-h hostname]

cibadmin (--cib_replace|-R)  -[Vrwlsmfbp]  [-i xml-object-id|
-o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
t-flag- whatever] [-h hostname]

cibadmin (--cib_update|-U) -[Vrwlsmfbp]  [-i xml-object-id|
-o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
t-flag- whatever] [-h hostname]

cibadmin (--cib_modify|-M) -[Vrwlsmfbp]  [-i xml-object-id|
-o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
t-flag- whatever] [-h hostname]

cibadmin (--cib_delete|-D) -[Vrwlsmfbp]  [-i xml-object-id|
-o xml-object-type] [-t t-flag-whatever] [-h hostname]

cibadmin (--cib_delete_alt|-d) -[Vrwlsmfbp] -o
xml-object-type [-X xml-string|-x xml-filename]
[-t t-flag-whatever]  [-h hostname]

cibadmin --cib_erase (-E)

cibadmin --cib_bump (-B)

cibadmin --cib_ismaster (-m)

cibadmin --cib_slave (-r)

cibadmin --cib_sync (-S)

cibadmin --cib_help (-?)
```

Description

The `cibadmin` command is the low-level administrative command for manipulating the Heartbeat CIB. Use it to dump all or part of the CIB, update all or part of it, modify all or part of it, delete the entire CIB, or perform miscellaneous CIB administrative operations.

`cibadmin` operates on the XML trees of the CIB, largely without knowledge of the meaning of the updates or queries performed. This means that shortcuts that seem natural to humans who understand the meaning of the elements in the XML tree are impossible to use with `cibadmin`. It requires a complete lack of ambiguity and can only deal with valid XML subtrees (tags and elements) for both input and output.

NOTE

`cibadmin` should always be used in preference to editing the `cib.xml` file by hand—especially if the cluster is active. The cluster goes to great lengths to detect and discourage this practice so that your data is not lost or corrupted.

Options

`--id` *`xml-object-id`*, `-i` *`xml-object-id`*
: Specify the XML ID of the XML object on which to operate.

  NOTE

  This option is deprecated and may be removed in future versions of `cibadmin`.

`--obj_type` *`object-type`*, `-o` *`object-type`*
: Specify the type of object on which to operate. Valid values are `nodes`, `resources`, `status`, and `constraints`.

`--verbose`, `-V`
: Turn on debug mode. Additional `-V` options increase the verbosity of the output.

`--help`, `-?`
: Obtain a help message from `cibadmin`.

`--cib_erase`, `-E`
: Erase the contents of the entire CIB.

`--cib_query`, `-Q`
: Query a portion of the CIB.

`--cib_create`, `-C`
: Create a new CIB from the XML content of the argument.

`--cib_replace`, `-R`
: Recursively replace an XML object in the CIB.

`--cib_update`, `-U`
: Recursively update an object in the CIB. Updating an object replaces like members in the XML, but does not delete attributes not mentioned.

`--cib_modify`, `-M`
: An alias for -U and --cib_update.

`--cib_delete`, `-D`
: Delete the first object matching the specified criteria, for example, `<tagname id="rsc1_op1" name="monitor"/>`. The tag name and all attributes must match for the element to be deleted.

`--cib_delete_alt`, `-d`
: Delete the object at the specified fully qualified location, for example, `<resource id="rsc1"><operations><op id="rsc1_op1"/>`. Requires `-o` *type* option.

`--cib_ismaster`, `-m`
: Print a message indicating whether or not the local instance of the CIB software is the master instance or not. Exits with return code 0 if it is the master instance or 35 if not.

`--cib_sync`, `-S`
: Force a resync of all nodes with the CIB on the specified host (if `-h` is used) or with the DC (if no `-h` option is used).

`--crm_xml` *`xml-fragment-string`* , `-X` *`xml-fragment-string`*
Specify an XML tag or fragment on which crmadmin should operate. It must be a complete tag or XML fragment.

`--xml-file` *`filename`*, `-x` *`filename`*
Specify XML from a file on which cibadmin should operate. It must be a complete tag or XML fragment.

`--xml_pipe`, `-p`
Specify that the XML on which `cibadmin` should operate comes from standard input. It must be a complete tag or XML fragment.

Specialized Options

`--cib_bump`, `-B`
Increase the `epoch` version counter in the CIB. Normally this value is increased automatically by the cluster when a new leader is elected. Manually increasing it can be useful if you want to make an older configuration obsolete (such as one stored on inactive cluster nodes).

`--cib_master`, `-w`
Force the local CIB instance into read-write mode. In normal circumstances, the cluster ensures that this is set correctly.

WARNING

This is a highly dangerous command in that it results in multiple instances of the CIB in read-write mode. Having more than one host respond to updates results in an inconsistent cluster.

`--cib_slave`, `-r`
Force the local CIB instance into read-only mode. In normal circumstances, the cluster ensures that this is set correctly.

WARNING

This is a highly dangerous command in that you may end up without any instance of the CIB in read-write mode. In this case, no instance would respond to requests that have not been sent to a specific node.

`--force_quorum`, `-f`
Force a write to the CIB regardless of whether the cluster has quorum.

IMPORTANT

Use this option with utmost care and avoid using it at a time when (part of) the cluster does not have quorum. Otherwise, this results in divergence between a small subset of the cluster nodes and the majority of the cluster. The worst case is if the majority of the cluster is also running (as in a *partitioned cluster*) and is also making changes. The problem arises when the complete cluster regains quorum and starts acting as one again. Before it can do so, it must pick the one configuration to continue to use. This is clearly problematic if there are significant configuration differences between the nodes.

`--host` *`hostname`*, `-h` *`hostname`*
Specify the host to which to send this command (rarely used, advanced option).

`--local`, `-l`
Let a command take effect locally (rarely used, advanced option).

`--no-bcast`, `-b`
Prevent the result of a command from being broadcast to other nodes even if it modifies the CIB.

IMPORTANT

Use this option with care to avoid ending up with a divergent cluster.

`--sync-call`, `-s`
Use the CIB's synchronous API when performing the operation given to `cibadmin`. This should have no noticeable effect for the user.

Examples

To get a copy of the entire active CIB (including status section, etc.) delivered to stdout, issue this command:

```
cibadmin -Q
```

To add an IPaddr2 resource to the *resources* section, first create a file `foo` with the following contents:

```
<primitive id="R_10.10.10.101" class="ocf" type="IPaddr2"
 provider="heartbeat">
 <instance_attributes id="RA_R_10.10.10.101">
  <attributes>
   <nvpair id="R_ip_P_ip" name="ip" value="10.10.10.101"/>
   <nvpair id="R_ip_P_nic" name="nic" value="eth0"/>
  </attributes>
 </instance_attributes>
</primitive>
```

Then issue the following command:

```
cibadmin --obj_type resources -U -x foo
```

To change the IP address of the IPaddr2 resource previously added, issue the command below:

```
cibadmin -M -X '<nvpair id="R_ip_P_ip" name="ip" value="10.10.10.102"/>'
```

NOTE

This does not change the resource name to match the new IP address. To do that, delete then re-add the resource with a new ID tag.

To stop (disable) the IP address resource added previously without removing it, create a file called `bar` with the following content in it:

```
<primitive id="R_10.10.10.101">
 <instance_attributes id="RA_R_10.10.10.101">
  <attributes>
   <nvpair id="stop_R_10.10.10.101" name="target_role" value="Stopped"/>
  </attributes>
 </instance_attributes>
</primitive>
```

Then issue the following command:

```
cibadmin --obj_type resources -U -x bar
```

To restart the IP address resource stopped by the previous step, issue:

```
cibadmin -D -X '<nvpair id="stop_R_10.10.10.101">'
```

To completely remove the IP address resource from the CIB, issue this command:

```
cibadmin -D -X '<primitive id="R_10.10.10.101"/>'
```

To replace the CIB with a new manually-edited version of the CIB, use the following command:

```
cibadmin -R -x $HOME/cib.xml
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk.

See Also

crm_resource(8) (page 89), crmadmin(8) (page 71), lrmadmin(8), heartbeat(8)

Author

`cibadmin` was written by Andrew Beekhof.

This manual page was originally written by Alan Robertson.

Caveats

Avoid working on the automatically maintained copy of the CIB on the local disk. Whenever anything in the cluster changes, the CIB is updated. Therefore using an outdated backup copy of the CIB to propagate your configuration changes might result in an inconsistent cluster.

crmadmin (8)

crmadmin — control the Cluster Resource Manager

Synopsis

```
crmadmin [-V|-q] [-i|-d|-K|-S|-E] node
crmadmin [-V|-q] -N -B
crmadmin [-V|-q] -D
crmadmin -v
crmadmin -?
```

Description

`crmadmin` was originally designed to control most of the actions of the CRM daemon. However, the largest part of its functionality has been made obsolete by other tools, such as `crm_attribute` and `crm_resource`. Its remaining functionality is mostly related to testing and the status of the crmd process.

WARNING

Some `crmadmin` options are geared towards testing and cause trouble if used incorrectly. In particular, do not use the `--kill` or `--election` options unless you know exactly what you are doing.

Options

`--help`, `-?`
: Print the help text.

`--version`, `-v`
: Print version details for HA, CRM, and CIB feature set.

`--verbose`, `-V`
: Turn on command debug information.

NOTE

Increase the level of verbosity by providing additional instances.

`--quiet`, `-q`
Do not provide any debug information at all and reduce the output to a minimum.

`--bash-export`, `-B`
Create bash export entries of the form `export uname=uuid`. This applies only to the `crmadmin -N node` command.

NOTE

The `-B` functionality is rarely useful and may be removed in future versions.

Commands

`--debug_inc node`, `-i node`
Increment the CRM daemon's debug level on the specified node. This can also be achieved by sending the USR1 signal to the crmd process.

`--debug_dec node`, `-d node`
Decrement the CRM daemon's debug level on the specified node. This can also be achieved by sending the USR2 signal to the crmd process.

`--kill node`, `-K node`
Shut down the CRM daemon on the specified node.

WARNING

Use this with extreme caution. This action should normally only be issued by Heartbeat and may have unintended side effects.

`--status node`, `-S node`
Query the status of the CRM daemon on the specified node.

The output includes a general heath indicator and the internal FSM state of the crmd process. This can be helpful when determining what the cluster is doing.

`--election` *node*, `-E` *node*
Initiate an election from the specified node.

WARNING

Use this with extreme caution. This action is normally initiated internally and may have unintended side effects.

`--dc_lookup`, `-D`
Query the uname of the current DC.

The location of the DC is only of significance to the crmd internally and is rarely useful to administrators except when deciding on which node to examine the logs.

`--nodes`, `-N`
Query the uname of all member nodes. The results of this query may include nodes in `offline` mode.

NOTE

The `-i`, `-d`, `-K`, and `-E` options are rarely used and may be removed in future versions.

See Also

crm_attribute(8) (page 74), crm_resource(8) (page 89)

Author

`crmadmin` was written by Andrew Beekhof.

crm_attribute (8)

crm_attribute — manipulate attributes in the CIB

Synopsis

```
crm_attribute [options]
```

Description

The `crm_attribute` command queries and manipulates node attributes and cluster configuration options that are used in the CIB.

Options

`--help`, `-?`
Print a help message.

`--verbose`, `-V`
Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
Retrieve rather than set the preference.

`--delete-attr`, `-D`
Delete rather than set the attribute.

`--attr-id` *string*, `-i` *string*
For advanced users only. Identifies the id attribute.

`--attr-value` *string*, `-v` *string*
Value to set. This is ignored when used with `-G`.

`--inhibit-policy-engine`, `-!`
For advanced users only.

`--node-uuid` *node_uuid*, `-u` *node_uuid*
Specify the UUID of the node to change.

`--node-uname` *node_uname*, `-U` *node_uname*
Specify the uname of the node to change.

`--set-name` *string*, `-s` *string*
Specify the set of attributes in which to read or write the attribute.

`--attr-name` *string*, `-n` *string*
Specify the attribute to set or query.

`--type` *string*, `-t` *type*
Determine to which section of the CIB the attribute should be set or to which section of the CIB the attribute that is queried belongs Possible values are `nodes`, `status`, or `crm_config`.

Additional Options:

Type	Additional Options
`nodes`	`-(U|u) -n [-s]`
`status`	`-(U|u) -n [-s]`
`crm_config`	`-n [-s]`

Examples

Query the value of the `location` attribute in the `nodes` section for the host *`myhost`* in the CIB:

```
crm_attribute -G -t nodes -U myhost -n location
```

Query the value of the `cluster-delay` attribute in the `crm_config` section in the CIB:

```
crm_attribute -G -t crm_config -n cluster-delay
```

Query the value of the `cluster-delay` attribute in the `crm_config` section in the CIB. Print just the value:

```
crm_attribute -G -Q -t crm_config -n cluster-delay
```

Delete the `location` attribute for the host *`myhost`* from the `nodes` section of the CIB:

```
crm_attribute -D -t nodes -U myhost -n location
```

Add a new attribute called `location` with the value of `office` to the *`set`* subsection of the `nodes` section in the CIB (settings applied to the host *`myhost`*):

```
crm_attribute -t nodes -U myhost -s set -n location -v office
```

Change the value of the `location` attribute in the `nodes` section for the *`myhost`* host:

```
crm_attribute -t nodes -U myhost -n location -v backoffice
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 64)

Author

`crm_attribute` was written by Andrew Beekhof.

crm_diff (8)

crm_diff — identify changes to the cluster configuration and apply patches to the configuration files

Synopsis

```
crm_diff [-?|-V] [-o filename] [-O string] [-p filename] [-n filename] [-N string]
```

Description

The `crm_diff` command assists in creating and applying XML patches. This can be useful for visualizing the changes between two versions of the cluster configuration or saving changes so they can be applied at a later time using `cibadmin`.

Options

`--help`, `-?`
Print a help message.

`--original` *filename*, `-o` *filename*
Specify the original file against which to diff or apply patches.

`--new` *filename*, `-n` *filename*
Specify the name of the new file.

`--original-string` *string*, `-O` *string*
Specify the original string against which to diff or apply patches.

`--new-string` *string*, `-N` *string*
Specify the new string.

`--patch` *filename*, `-p` *filename*
Apply a patch to the original XML. Always use with `-o`.

`--cib, -c`
: Compare or patch the inputs as a CIB. Always specify the base version with `-o` and provide either the patch file or the second version with `-p` or `-n`, respectively.

`--stdin, -s`
: Read the inputs from stdin.

Examples

Use crm_diff to determine the differences between various CIB configuration files and to create patches. By means of patches, easily reuse configuration parts without having to use the cibadmin command on every single one of them.

1. Obtain the two different configuration files by running cibadmin on the two cluster setups to compare:

   ```
   cibadmin -Q > cib1.xml
   cibadmin -Q > cib2.xml
   ```

2. Determine whether to diff the entire files against each other or compare just a subset of the configurations.

3. To print the difference between the files to stdout, use the following command:

   ```
   crm_diff -o cib1.xml -n cib2.xml
   ```

4. To print the difference between the files to a file and create a patch, use the following command:

   ```
   crm_diff -o cib1.xml -n cib2.xml > patch.xml
   ```

5. Apply the patch to the original file:

   ```
   crm_diff -o cib1.xml -p patch.xml
   ```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 64)

Author

`crm_diff` was written by Andrew Beekhof.

crm_failcount (8)

crm_failcount — manipulate the failcount attribute on a given resource

Synopsis

```
crm_failcount [-?|-V] -D -u|-U node -r resource
crm_failcount [-?|-V] -G -u|-U node -r resource
crm_failcount [-?|-V] -v string -u|-U node -r resource
```

Description

Heartbeat implements a sophisticated method to compute and force failover of a resource to another node in case that resource tends to fail on the current node. A resource carries a `resource_stickiness` attribute to determine how much it prefers to run on a certain node. It also carries a `resource_failure_stickiness` that determines the threshold at which the resource should failover to another node.

The `failcount` attribute is added to the resource and increased on resource monitoring failure. The value of `failcount` multiplied by the value of `resource_failure_stickiness` determines the *failover score* of this resource. If this number exceeds the preference set for this resource, the resource is moved to another node and not run again on the original node until the failure count is reset.

The `crm_failcount` command queries the number of failures per resource on a given node. This tool can also be used to reset the failcount, allowing the resource to run again on nodes where it had failed too often.

Options

`--help`, `-?`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

NOTE

Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
Retrieve rather than set the preference.

`--delete-attr`, `-D`
Specify the attribute to delete.

`--attr-id` *`string`*, `-i` *`string`*
For advanced users only. Identifies the id attribute.

`--attr-value` *`string`*, `-v` *`string`*
Specify the value to use. This option is ignored when used with `-G`.

`--node-uuid` *`node_uuid`*, `-u` *`node_uuid`*
Specify the UUID of the node to change.

`--node-uname` *`node_uname`*, `-U` *`node_uname`*
Specify the uname of the node to change.

`--resource-id` *`resource name`*, `-r` *`resource name`*
Specify the name of the resource on which to operate.

`--inhibit-policy-engine`, `-!`
For advanced users only.

Examples

Reset the failcount for the resource `myrsc` on the node `node1`:

```
crm_failcount -D -U node1 -r my_rsc
```

Query the current failcount for the resource `myrsc` on the node `node1`:

```
crm_failcount -G -U node1 -r my_rsc
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

crm_attribute(8) (page 74), cibadmin(8) (page 64), and the Linux High Availability FAQ Web site [http://www.linux-ha.org/v2/faq/forced_failover]

Author

`crm_failcount` was written by Andrew Beekhof.

crm_master (8)

crm_master — determine which resource instance to promote to master

Synopsis

```
crm_master [-V|-Q] -D [-l lifetime]
crm_master [-V|-Q] -G [-l lifetime]
crm_master [-V|-Q] -v string [-l string]
```

Description

`crm_master` is called from inside the resource agent scripts to determine which resource instance should be promoted to master mode. It should never be used from the command line and is just a helper utility for the resource agents. RAs use `crm_master` to promote a particular instance to master mode or to remove this preference from it. By assigning a lifetime, determine whether this setting should survive a reboot of the node (set lifetime to `forever`) or whether it should not survive a reboot (set lifetime to `reboot`).

A resource agent needs to determine on which resource `crm_master` should operate. These queries must be handled inside the resource agent script. The actual calls of `crm_master` follow a syntax similar to those of the `crm_attribute` command.

Options

`--help`, `-?`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
: Retrieve rather than set the preference to be promoted.

`--delete-attr`, `-D`
: Delete rather than set the attribute.

`--attr-id` *string*, `-i` *string*
: For advanced users only. Identifies the id attribute.

`--attr-value` *string*, `-v` *string*
: Value to set. This is ignored when used with `-G`.

`--lifetime` *string*, `-l` *string*
: Specify how long the preference lasts. Possible values are `reboot` or `forever`.

Environment Variables

`OCF_RESOURCE_INSTANCE`—the name of the resource instance

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk.

See Also

cibadmin(8) (page 64), crm_attribute(8) (page 74)

Author

crm_master was written by Andrew Beekhof.

crm_mon (8)

crm_mon — monitor the cluster's status

Synopsis

```
crm_mon [-V] -d -pfilename -h filename
crm_mon [-V] [-1|-n|-r] -h filename
crm_mon [-V] [-n|-r] -X filename
crm_mon [-V] [-n|-r] -c|-1
crm_mon [-V] -i interval
crm_mon -?
```

Description

The `crm_mon` command allows you to monitor your cluster's status and configuration. Its output includes the number of nodes, uname, uuid, status, the resources configured in your cluster, and the current status of each. The output of `crm_mon` can be displayed at the console or printed into an HTML file. When provided with a cluster configuration file without the status section, `crm_mon` creates an overview of nodes and resources as specified in the file.

Options

`--help`, `-?`
: Provide help.

`--verbose`, `-V`
: Increase the debug output.

`--interval` *seconds*, `-i` *seconds*
: Determine the update frequency. If `-i` is not specified, the default of 15 seconds is assumed.

`--group-by-node`, `-n`
: Group resources by node.

`--inactive`, `-r`
: Display inactive resources.

`--as-console`, `-c`
: Display the cluster status on the console.

`--simple-status`, `-s`
: Display the cluster status once as a simple one line output (suitable for nagios).

`--one-shot`, `-1`
: Display the cluster status once on the console then exit (does not use ncurses).

`--as-html` *`filename`*, `-h` *`filename`*
: Write the cluster's status to the specified file.

`--web-cgi`, `-w`
: Web mode with output suitable for CGI.

`--daemonize`, `-d`
: Run in the background as a daemon.

`--pid-file` *`filename`*, `-p` *`filename`*
: Specify the daemon's pid file.

Examples

Display your cluster's status and get an updated listing every 15 seconds:

```
crm_mon
```

Display your cluster's status and get an updated listing after an interval specified by `-i`. If `-i` is not given, the default refresh interval of 15 seconds is assumed:

```
crm_mon -i interval[s]
```

Display your cluster's status on the console:

```
crm_mon -c
```

Display your cluster's status on the console just once then exit:

```
crm_mon -1
```

Display your cluster's status and group resources by node:

```
crm_mon -n
```

Display your cluster's status, group resources by node, and include inactive resources in the list:

```
crm_mon -n -r
```

Write your cluster's status to an HTML file:

```
crm_mon -h filename
```

Run `crm_mon` as a daemon in the background, specify the daemon's pid file for easier control of the daemon process, and create HTML output. This option allows you to constantly create HTML output that can be easily processed by other monitoring applications:

```
crm_mon -d -p filename -h filename
```

Display the cluster configuration laid out in an existing cluster configuration file (*`filename`*), group the resources by node, and include inactive resources. This command can be used for dry-runs of a cluster configuration before rolling it out to a live cluster.

```
crm_mon -r -n -X filename
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

Author

`crm_mon` was written by Andrew Beekhof.

crm_resource (8)

crm_resource — interact with the Cluster Resource Manager

Synopsis

```
crm_resource  [-?|-V|-S] -L|-Q|-W|-D|-C|-P|-p [options]
```

Description

The `crm_resource` command performs various resource-related actions on the cluster. It can modify the definition of configured resources, start and stop resources, and delete and migrate resources between nodes.

`--help`, `-?`
Print the help message.

`--verbose`, `-V`
Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
Print only the value on stdout (for use with `-W`).

Commands

`--list`, `-L`
List all resources.

`--query-xml`, `-x`
Query a resource.

Requires: `-r`

`--locate`, `-W`
Locate a resource.

Requires: `-r`

`--migrate`, `-M`
Migrate a resource from its current location.

Use `-H` to specify a destination.

If `-H` is not specified, the resource is forced to move by creating a rule for the current location and a score of `-INFINITY`.

NOTE

This prevents the resource from running on this node until the constraint is removed with `-U`.

Requires: `-r`, Optional: `-H`, `-f`

`--un-migrate`, `-U`
Remove all constraints created by `-M`

Requires: `-r`

`--delete`, `-D`
Delete a resource from the CIB.

Requires: `-r`, `-t`

`--cleanup`, `-C`
Delete a resource from the LRM.

Requires: `-r`. Optional: `-H`

`--reprobe`, `-P`
Recheck for resources started outside the CRM.

Optional: `-H`

`--refresh`, `-R`
Refresh the CIB from the LRM.

Optional: `-H`

`--set-parameter` *string*, `-p` *string*
Set the named parameter for a resource.

Requires: `-r`, `-v`. Optional: `-i`, `-s`

`--get-parameter` *string*, `-g` *string*
Get the named parameter for a resource.

Requires: `-r`. Optional: `-i`, `-s`

`--delete-parameter` *string*, `-d` *string*
Delete the named parameter for a resource.

Requires: `-r`. Optional: `-i`

Options

`--resource` *string*, `-r` *string*
Specify the resource ID.

`--resource-type` *string*, `-t` *string*
Specify the resource type (`primitive`, `clone`, `group`, etc.).

`--property-value` *string*, `-v` *string*
Specify the property value.

`--host-uname` *string*, `-H` *string*
Specify the hostname.

`--meta`
Modify a resource's configuration option rahter than one which is passed to the resouce agent script. For use with `-p`, `-g` and `-d`.

`--lifetime` *`string`*, `-u` *`string`*
Lifespan of migration constraints.

`--force`, `-f`
Force the resource to move by creating a rule for the current location and a score of `-INFINITY`

This should be used if the resource's stickiness and constraint scores total more than `INFINITY` (currently 100,000).

NOTE

This prevents the resource from running on this node until the constraint is removed with `-U`.

`-s` *`string`*
(Advanced Use Only) Specify the ID of the `instance_attributes` object to change.

`-i` *`string`*
(Advanced Use Only) Specify the ID of the `nvpair` object to change or delete.

Examples

Listing all resources:

```
crm_resource -L
```

Checking where a resource is running (and if it is):

```
crm_resource -W -r my_first_ip
```

If the `my_first_ip` resource is running, the output of this command reveals the node on which it is running. If it is not running, the output shows this.

Start or stop a resource:

```
crm_resource -r my_first_ip -p target_role -v started
crm_resource -r my_first_ip -p target_role -v stopped
```

Query the definition of a resource:

```
crm_resource -Q -r my_first_ip
```

Migrate a resource away from its current location:

```
crm_resource -M -r my_first_ip
```

Migrate a resource to a specific location:

```
crm_resource -M -r my_first_ip -H c001n02
```

Allow a resource to return to its normal location:

```
crm_resource -U -r my_first_ip
```

NOTE

The values of `resource_stickiness` and `default_resource_stickiness` may mean that it does not move back. In such cases, you should use `-M` to move it back before running this command.

Delete a resource from the CRM:

```
crm_resource -D -r my_first_ip -t primitive
```

Delete a resource group from the CRM:

```
crm_resource -D -r my_first_group -t group
```

Disable resource management for a resource in the CRM:

```
crm_resource -p is_managed -r my_first_ip -t primitive -v off
```

Enable resource management for a resource in the CRM:

```
crm_resource -p is_managed -r my_first_ip -t primitive -v on
```

Reset a failed resource after having been manually cleaned up:

```
crm_resource -C -H c001n02 -r my_first_ip
```

Recheck all nodes for resources started outside the CRM:

```
crm_resource -P
```

Recheck one node for resources started outside the CRM:

```
crm_resource -P -H c001n02
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 64), crmadmin(8) (page 71), lrmadmin(8), heartbeat(8)

Author

`crm_resource` was written by Andrew Beekhof.

crm_standby (8)

crm_standby — manipulate a node's standby attribute to determine whether resources can be run on this node

Synopsis

```
crm_standby [-?|-V] -D -u|-U node -r resource

crm_standby  [-?|-V] -G -u|-U node -r resource

crm_standby [-?|-V] -v string -u|-U node -r resource [-l string]
```

Description

The `crm_standby` command manipulates a node's standby attribute. Any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Standby mode can be useful for performing maintenance tasks, such as kernel updates. Remove the standby attribute from the node when it should become a fully active member of the cluster again.

By assigning a lifetime to the `standby` attribute, determine whether the standby setting should survive a reboot of the node (set lifetime to `forever`) or should be reset with reboot (set lifetime to `reboot`). Alternatively, remove the `standby` attribute and bring the node back from standby manually.

Options

`--help`, `-?`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--quiet`, `-Q`
: When doing an attribute query using `-G`, print just the value to stdout. Use this option with `-G`.

`--get-value`, `-G`
: Retrieve rather than set the preference.

`--delete-attr`, `-D`
: Specify the attribute to delete.

`--attr-value` *`string`*, `-v` *`string`*
: Specify the value to use. This option is ignored when used with `-G`.

`--attr-id` *`string`*, `-i` *`string`*
: For advanced users only. Identifies the id attribute..

`--node-uuid` *`node_uuid`*, `-u` *`node_uuid`*
: Specify the UUID of the node to change.

`--node-uname` *`node_uname`*, `-U` *`node_uname`*
: Specify the uname of the node to change.

`--lifetime` *`string`*, `-l` *`string`*
: Determine how long this preference lasts. Possible values are `reboot` or `forever`.

NOTE

If a `forever` value exists, it is always used by the CRM instead of any `reboot` value.

Examples

Have a local node go to standby:

```
crm_standby -v true
```

Have a node (`node1`) go to standby:

```
crm_standby -v true -U node1
```

Query the standby status of a node:

```
crm_standby -G -U node1
```

Remove the standby property from a node:

```
crm_standby -D -U node1
```

Have a node go to standby for an indefinite period of time:

```
crm_standby -v true -l forever -U node1
```

Have a node go to standby until the next reboot of this node:

```
crm_standby -v true -l reboot -U node1
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 64), crm_attribute(8) (page 74)

Author

`crm_standby` was written by Andrew Beekhof.

crm_uuid (8)

crm_uuid — get a node's UUID

Synopsis

```
crm_uuid [-w|-r]
```

Description

UUIDs are used to identify cluster nodes to ensure that they can always be uniquely identified. The `crm_uuid` command displays and modifies the UUID of the node on which it is run.

When Heartbeat is first started on a node, it creates a UUID (in binary form) in `/var/lib/heartbeat/hb_uuid`. This file can be read and written by means of `crm_uuid`.

NOTE

There are very rare circumstances when `crm_uuid` should be used to modify the UUID file. The most common is when it is necessary to set a node's UUID to a known value when creating a new cluster.

Options

`--write`, `-w`

Write a UUID value to the `/var/lib/heartbeat/hb_uuid` file.

WARNING

Use the `-w` option with care, because it creates a new UUID value for the node on which it is run. Various scripts across the cluster might still use the old value, causing the cluster to fail because the node is no longer as-

sociated with the old UUID value. Do not change the UUID unless you changed all references to it as well.

`--read`, `-r`
Read the UUID value and print it to stdout.

See Also

`/var/lib/heartbeat/hb_uuid`

Author

`crm_uuid` was written by Andrew Beekhof.

crm_verify (8)

crm_verify — check the CIB for consistency

Synopsis

```
crm_verify [-V] -x file
crm_verify [-V] -X string
crm_verify [-V] -L|-p
crm_verify [-?]
```

Description

`crm_verify` checks the configuration database (CIB) for consistency and other problems. It can be used to check a file containing the configuration or can it can connect to a running cluster. It reports two classes of problems, errors and warnings. Errors must be fixed before Heartbeat can work properly. However, it is left up to the administrator to decide if the warnings should also be fixed.

`crm_verify` assists in creating new or modified configurations. You can take a local copy of a CIB in the running cluster, edit it, validate it using `crm_verify`, then put the new configuration into effect using `cibadmin`.

Options

`--help`, `-h`
: Print a help message.

`--verbose`, `-V`
: Turn on debug information.

> **NOTE**
>
> Increase the level of verbosity by providing additional instances.

`--live-check`, `-L`
Connect to the running cluster and check the CIB.

`--crm_xml` *`string`*, `-X` *`string`*
Check the configuration in the supplied string. Pass complete CIBs only.

`--xml-file` *`file`*, `-x` *`file`*
Check the configuration in the named file.

`--xml-pipe`, `-p`
Use the configuration piped in via stdin. Pass complete CIBs only.

`--dtd-file` *`string`*, `-D` *`string`*
Use the given DTD file instead of `/usr/share/heartbeat/crm.dtd`.

Examples

Check the consistency of the configuration in the running cluster and produce verbose output:

```
crm_verify -VL
```

Check the consistency of the configuration in a given file and produce verbose output:

```
crm_verify -Vx file1
```

Pipe a configuration into `crm_verify` and produce verbose output:

```
cat file1.xml | crm_verify -Vp
```

Files

`/var/lib/heartbeat/crm/cib.xml`—the CIB (minus status section) on disk. Editing this file directly is strongly discouraged.

See Also

cibadmin(8) (page 64)

Author

`crm_verify` was written by Andrew Beekhof.

7 Creating Resources

All tasks that should be managed by a cluster must be available as a resource. There are two major groups that should be distinguished: resource agents and STONITH agents. For both categories, you can add your own agents, extending the abilities of the cluster to your own needs.

7.1 STONITH Agents

A cluster sometimes detects that one of the nodes is misbehaving and needs to remove it. This is called *fencing* and is commonly done with a STONITH resource. All STONITH resources reside in `/usr/lib/stonith/plugins` on each node. Exactly how the agent fences the node varies. Methods include powering off, rebooting, and shutting down.

To test a configuration, it is sufficient to use the `ssh` STONITH agent, which simply powers off the node. However, because it is impossible to know how SSH might react to other system problems, this STONITH agent is not a good choice for a production environment.

To get a list of all currently available STONITH devices (from the software side), use the command `stonith -L`.

Unfortunately, there is no documentation about writing STONITH agents yet. If you want to write new STONITH agents, consult the examples available in the source of the heartbeat package.

7.2 Resource Agents

All services that are controlled by the cluster have a corresponding RA that handles the changes and monitoring of this service. These RAs are available in three different flavors:

Heartbeat 1 RAs
: All the resources from Heartbeat 1 are still available in Heartbeat 2. However, it is recommended to migrate your configurations to Heartbeat 2 OCF RAs if possible.

LSB Scripts
: LSB scripts are exactly the same scripts as used in the initialization process when booting the system. Read more about these scripts in the *Installation and Administration* guide of SUSE Linux Enterprise Server. If you are familiar with this kind of script, it may be the easiest way to extend Heartbeat to your needs.

OCF RA Scripts
: OCF RA scripts are best suited for use with Heartbeat. With these scripts, it is also possible to add the same service type multiple times and still control each of these resources individually. There is also a standard interface for the documentation of this kind of RA scripts. In some cases, such as when using `master_slave` resources or when needing special monitoring abilities, you need this type of resource.

For a detailed list of all available OCF RAs, refer to Appendix A, *HB OCF Agents* (page 109).

7.3 Writing OCF Resource Agents

All OCF RAs are available in `/usr/lib/ocf/resource.d/`. The scripts that are delivered with the Heartbeat package are found in this directory in the subdirectory `heartbeat`. To avoid name clashes, create a different subdirectory when creating new resource agents. For example, if you have a resource group `kitchen` with the resource `coffee_machine`, add this resource to the directory `/usr/lib/ocf/resource.d/kitchen/`. To access this RA, the configuration would look like:

```
<primitive id="coffee_1" class="ocf" type="coffee_machine" provider="kitchen"/>
```

When implementing your own OCF RA, provide several actions for this agent. More details about writing OCF resource agents can be found at http://www.linux-ha.org/OCFResourceAgent. Find special information about several concepts of Heartbeat 2 at http://linux-ha.org/v2/Concepts.

8

Troubleshooting

Especially when starting to experiment with Heartbeat, strange problems may occur that are not easy to understand. However, there are several utilities that may be used to take a closer look at the Heartbeat internal processes.

What is the state of my cluster?

To check the current state of your cluster, use the program `crm_mon`. This displays the current DC as well as all of the nodes and resources that are known to the current node.

Several nodes of my cluster do not see each other.

For some reason, the connection between your nodes is broken. Most often, this is the result of a badly configured firewall. This also may be the reason for a *split brain* condition, where the cluster is partitioned.

I want to list my currently known resources.

Use the command `crm_resource -L` to learn about your current resources.

I configured a resource, but it always fails.

Try to run the resource agent manually. With LSB, just run *`scriptname`* `start` and *`scriptname`* `stop`. To check an OCF script, set the needed environment variables first. For example, when testing the `IPaddr` OCF script, you have to set the value for the variable `ip` by setting an environment variable that prefixes the name of the variable with `OCF_RESKEY_`. For this example, run the command:

```
export OCF_RESKEY_ip=<your_ip_address>
/usr/lib/ocf/resource.d/heartbeat/IPaddr validate-all
/usr/lib/ocf/resource.d/heartbeat/IPaddr start
/usr/lib/ocf/resource.d/heartbeat/IPaddr stop
```

If this fails, it is very likely that you missed some mandatory variable or just mistyped a parameter.

I just get a failed message. Is it possible to get more information?

You may always add the `-V` parameter to your commands. If you do that multiple times, the debug output becomes very verbose.

How can I clean up my resources?

If you know the IDs of your resources, which you can get with `crm_resource -L`, remove a specific resource with `crm_resource -C -r` *`resource id`*.

For additional information about high availability on Linux and Heartbeat including configuring cluster resources and managing and customizing a Heartbeat cluster, see http://www.linu-ha.org.

HB OCF Agents

A

All OCF agents require several parameters to be set when they are started. The following overview shows how to manually operate these agents. The data that is available in this appendix is directly taken from the `meta-data` invocation of the respective RA. Find all these agents in `/usr/lib/ocf/resource.d/heartbeat/`.

When configuring an RA, omit the `OCF_RESKEY_` prefix to the parameter name. Parameters that are in square brackets may be omitted in the configuration.

ocf:apache (7)

ocf:apache — Apache web server

Synopsis

OCF_RESKEY_configfile=string [OCF_RESKEY_httpd=string] [OCF_RESKEY_port=integer] [OCF_RESKEY_statusurl=string] [OCF_RESKEY_options=string] [OCF_RESKEY_testregex=string] apache [start | stop | status | monitor | meta-data | validate-all]

Description

This is the resource agent for the Apache web server. Thie resource agent operates both version 1.x and version 2.x Apache servers. See also http://httpd.apache.org/

Supported Parameters

OCF_RESKEY_configfile=configuration file path
The full pathname of the Apache configuration file. This file is parsed to provide defaults for various other resource agent parameters.

OCF_RESKEY_httpd=httpd binary path
The full pathname of the httpd binary (optional).

OCF_RESKEY_port=httpd port
A port number that we can probe for status information using the statusurl. This will default to the port number found in the configuration file, or 80, if none can be found in the configuration file.

OCF_RESKEY_statusurl=url name
The URL of the apache status module. If left unspecified, it will be inferred from the apache configuration file.

`OCF_RESKEY_options`=command line options
Extra options to apply when starting apache. See man httpd(8).

`OCF_RESKEY_testregex`=test regular expression
Regular expression to match in the output of statusurl. It is case insensitive.

ocf:AudibleAlarm (7)

ocf:AudibleAlarm — AudibleAlarm resource agent

Synopsis

[OCF_RESKEY_nodelist=string] AudibleAlarm [start | stop | restart | status | monitor | meta-data | validate-all]

Description

Resource script for AudibleAlarm. It sets an audible alarm running by beeping at a set interval.

Supported Parameters

OCF_RESKEY_nodelist=Node list

The node list that should never sound the alarm.

ocf:ClusterMon (7)

ocf:ClusterMon — ClusterMon resource agent

Synopsis

[OCF_RESKEY_user=string] [OCF_RESKEY_update=integer]
[OCF_RESKEY_extra_options=string] OCF_RESKEY_pidfile=string
OCF_RESKEY_htmlfile=string ClusterMon [start | stop | monitor | meta-data | validate-all]

Description

This is a ClusterMon Resource Agent. It outputs current cluster status to the html.

Supported Parameters

OCF_RESKEY_user=The user we want to run crm_mon as
The user we want to run crm_mon as

OCF_RESKEY_update=Update interval
How frequently should we update the cluster status

OCF_RESKEY_extra_options=Extra options
Additional options to pass to crm_mon. Eg. -n -r

OCF_RESKEY_pidfile=PID file
PID file location to ensure only one instance is running

OCF_RESKEY_htmlfile=HTML output
Location to write HTML output to.

ocf:db2 (7)

ocf:db2 — db2 resource agent

Synopsis

[OCF_RESKEY_instance=string] [OCF_RESKEY_admin=string] db2 [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for db2. It manages a DB2 Universal Database instance as an HA resource.

Supported Parameters

OCF_RESKEY_instance=instance
The instance of database.

OCF_RESKEY_admin=admin
The admin user of the instance.

ocf:Delay (7)

ocf:Delay — Delay resource agent

Synopsis

[OCF_RESKEY_startdelay=integer] [OCF_RESKEY_stopdelay=integer] [OCF_RESKEY_mondelay=integer] Delay [start | stop | status | monitor | meta-data | validate-all]

Description

This script is a test resource for introducing delay.

Supported Parameters

OCF_RESKEY_startdelay=Start delay
How long in seconds to delay on start operation.

OCF_RESKEY_stopdelay=Stop delay
How long in seconds to delay on stop operation. Defaults to "startdelay" if unspecified.

OCF_RESKEY_mondelay=Monitor delay
How long in seconds to delay on monitor operation. Defaults to "startdelay" if unspecified.

ocf:drbd (7)

ocf:drbd — This resource agent manages a Distributed Replicated Block Device (DRBD) object as a master/slave resource. DRBD is a mechanism for replicating storage; please see the documentation for setup details.

Synopsis

OCF_RESKEY_drbd_resource=string [OCF_RESKEY_drbdconf=string] [OCF_RESKEY_clone_overrides_hostname=boolean] [OCF_RESKEY_clone_max=integer] [OCF_RESKEY_clone_node_max=integer] [OCF_RESKEY_master_max=integer] [OCF_RESKEY_master_node_max=integer] drbd [start | promote | demote | notify | stop | monitor | monitor | meta-data | validate-all]

Description

Master/Slave OCF Resource Agent for DRBD

Supported Parameters

OCF_RESKEY_drbd_resource=drbd resource name
The name of the drbd resource from the drbd.conf file.

OCF_RESKEY_drbdconf=Path to drbd.conf
Full path to the drbd.conf file.

OCF_RESKEY_clone_overrides_hostname=Override drbd hostname
Whether or not to override the hostname with the clone number. This can be used to create floating peer configurations; drbd will be told to use node_<cloneno> as the hostname instead of the real uname, which can then be used in drbd.conf.

OCF_RESKEY_clone_max=Number of clones
Number of clones of this drbd resource. Do not fiddle with the default.

`OCF_RESKEY_clone_node_max`=Number of nodes
: Clones per node. Do not fiddle with the default.

`OCF_RESKEY_master_max`=Number of primaries
: Maximum number of active primaries. Do not fiddle with the default.

`OCF_RESKEY_master_node_max`=Number of primaries per node
: Maximum number of primaries per node. Do not fiddle with the default.

ocf:Dummy (7)

ocf:Dummy — Dummy resource agent

Synopsis

`OCF_RESKEY_state`=string `Dummy` [start | stop | monitor | reload | migrate_to | migrate_from | meta-data | validate-all]

Description

This is a Dummy Resource Agent. It does absolutely nothing except keep track of whether it is running or not. Its function is to test and to serve as a template for RA writers.

Supported Parameters

`OCF_RESKEY_state`=State file

Location to store the resource state in.

ocf:eDir88 (7)

ocf:eDir88 — eDirectory resource agent

Synopsis

OCF_RESKEY_eDir_config_file=string
[OCF_RESKEY_eDir_monitor_ldap=boolean]
[OCF_RESKEY_eDir_monitor_idm=boolean]
[OCF_RESKEY_eDir_jvm_initial_heap=integer]
[OCF_RESKEY_eDir_jvm_max_heap=integer]
[OCF_RESKEY_eDir_jvm_options=string] eDir88 [start | stop | monitor | meta-data | validate-all]

Description

Resource script for managing an eDirectory instance. Manages a single instance of eDirectory as an HA resource. The "multiple instances" feature or eDirectory has been added in version 8.8. This script will not work for any version of eDirectory prior to 8.8. This RA can be used to load multiple eDirectory instances on the same host. It is very strongly recommended to put eDir configuration files (as per the eDir_config_file parameter) on local storage on each node. This is necessary for this RA to be able to handle situations where the shared storage has become unavailable. If the eDir configuration file is not available, this RA will fail, and heartbeat will be unable to manage the resource. Side effects include STONITH actions, unmanageable resources, etc... Setting a high action timeout value is _very_ _strongly_ recommended. eDir with IDM can take in excess of 10 minutes to start. If heartbeat times out before eDir has had a chance to start properly, mayhem _WILL ENSUE_. The LDAP module seems to be one of the very last to start. So this script will take even longer to start on installations with IDM and LDAP if the monitoring of IDM and/or LDAP is enabled, as the start command will wait for IDM and LDAP to be available.

Supported Parameters

`OCF_RESKEY_eDir_config_file`=eDir config file
Path to configuration file for eDirectory instance.

`OCF_RESKEY_eDir_monitor_ldap`=eDir monitor ldap
Should we monitor if LDAP is running for the eDirectory instance?

`OCF_RESKEY_eDir_monitor_idm`=eDir monitor IDM
Should we monitor if IDM is running for the eDirectory instance?

`OCF_RESKEY_eDir_jvm_initial_heap`=DHOST_INITIAL_HEAP value
Value for the DHOST_INITIAL_HEAP java environment variable. If unset, java defaults will be used.

`OCF_RESKEY_eDir_jvm_max_heap`=DHOST_MAX_HEAP value
Value for the DHOST_MAX_HEAP java environment variable. If unset, java defaults will be used.

`OCF_RESKEY_eDir_jvm_options`=DHOST_OPTIONS value
Value for the DHOST_OPTIONS java environment variable. If unset, original values will be used.

ocf:Evmsd (7)

ocf:Evmsd — Evmsd resource agent

Synopsis

`Evmsd` [start | stop | monitor | meta-data]

Description

This is a Evmsd Resource Agent.

Supported Parameters

ocf:EvmsSCC (7)

ocf:EvmsSCC — EVMS SCC resource agent

Synopsis

`EvmsSCC` [start | stop | notify | status | monitor | meta-data]

Description

Resource script for EVMS shared cluster container. It runs evms_activate on one node in the cluster.

Supported Parameters

ocf:Filesystem (7)

ocf:Filesystem — Filesystem resource agent

Synopsis

[OCF_RESKEY_device=string] [OCF_RESKEY_directory=string]
[OCF_RESKEY_fstype=string] [OCF_RESKEY_options=string]
[OCF_RESKEY_ocfs2_cluster=string]
[OCF_RESKEY_ocfs2_configfs=string] Filesystem [start | stop | notify | monitor | validate-all | meta-data]

Description

Resource script for Filesystem. It manages a Filesystem on a shared storage medium.

Supported Parameters

OCF_RESKEY_device=block device
: The name of block device for the filesystem, or -U, -L options for mount, or NFS mount specification.

OCF_RESKEY_directory=mount point
: The mount point for the filesystem.

OCF_RESKEY_fstype=filesystem type
: The optional type of filesystem to be mounted.

OCF_RESKEY_options=options
: Any extra options to be given as -o options to mount.

OCF_RESKEY_ocfs2_cluster=OCFS2 cluster name/UUID
: The name (UUID) of the OCFS2 cluster this filesystem is part of, iff this is an OCFS2 resource and there's more than one cluster. You should not need to specify this.

`OCF_RESKEY_ocfs2_configfs`=OCFS2 configfs root
: Mountpoint of the cluster hierarchy below configfs. You should not need to specify this.

ocf:ICP (7)

ocf:ICP — ICP resource agent

Synopsis

[OCF_RESKEY_driveid=string] [OCF_RESKEY_device=string] ICP [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for ICP. It Manages an ICP Vortex clustered host drive as an HA resource.

Supported Parameters

OCF_RESKEY_driveid=ICP cluster drive ID
The ICP cluster drive ID.

OCF_RESKEY_device=device
The device name.

ocf:ids (7)

ocf:ids — OCF resource agent for the IBM's database server called Informix Dynamic Server (IDS)

Synopsis

[OCF_RESKEY_informixdir=string] [OCF_RESKEY_informixserver=string] [OCF_RESKEY_onconfig=string] [OCF_RESKEY_dbname=string] [OCF_RESKEY_sqltestquery=string] ids [start | stop | status | monitor | validate-all | meta-data | methods | usage]

Description

OCF resource agent to manage an IBM Informix Dynamic Server (IDS) instance as an High-Availability resource.

Supported Parameters

OCF_RESKEY_informixdir= INFORMIXDIR environment variable
The value the environment variable INFORMIXDIR has after a typical installation of IDS. Or in other words: the path (without trailing '/') where IDS was installed to. If this parameter is unspecified the script will try to get the value from the shell environment.

OCF_RESKEY_informixserver= INFORMIXSERVER environment variable
The value the environment variable INFORMIXSERVER has after a typical installation of IDS. Or in other words: the name of the IDS server instance to manage. If this parameter is unspecified the script will try to get the value from the shell environment.

OCF_RESKEY_onconfig= ONCONFIG environment variable
The value the environment variable ONCONFIG has after a typical installation of IDS. Or in other words: the name of the configuration file for the IDS instance specified in INFORMIXSERVER. The specified configuration file will be searched

at '/etc/'. If this parameter is unspecified the script will try to get the value from the shell environment.

`OCF_RESKEY_dbname`= database to use for monitoring, defaults to 'sysmaster'
This parameter defines which database to use in order to monitor the IDS instance. If this parameter is unspecified the script will use the 'sysmaster' database as a default.

`OCF_RESKEY_sqltestquery`= SQL test query to use for monitoring, defaults to 'SELECT COUNT(*) FROM systables;'
SQL test query to run on the database specified by the parameter 'dbname' in order to monitor the IDS instance and determine if it's functional or not. If this parameter is unspecified the script will use 'SELECT COUNT(*) FROM systables;' as a default.

ocf:IPaddr2 (7)

ocf:IPaddr2 — Manages virtual IPv4 addresses

Synopsis

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string]
[OCF_RESKEY_cidr_netmask=string] [OCF_RESKEY_broadcast=string]
[OCF_RESKEY_iflabel=string] [OCF_RESKEY_lvs_support=boolean]
[OCF_RESKEY_mac=string] [OCF_RESKEY_clusterip_hash=string]
[OCF_RESKEY_arp_interval=integer] [OCF_RESKEY_arp_count=integer]
[OCF_RESKEY_arp_bg=string] [OCF_RESKEY_arp_mac=string] IPaddr2 [start
| stop | status | monitor | meta-data | validate-all]

Description

This Linux-specific resource manages IP alias IP addresses. It can add an IP alias, or remove one. In addition, it can implement Cluster Alias IP functionality if invoked as a clone resource.

Supported Parameters

OCF_RESKEY_ip=IPv4 address
: The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1".

OCF_RESKEY_nic=Network interface
: The base network interface on which the IP address will be brought online. If left empty, the script will try and determine this from the routing table. Do NOT specify an alias interface in the form eth0:1 or anything here; rather, specify the base interface only.

OCF_RESKEY_cidr_netmask=CIDR netmask
: The netmask for the interface in CIDR format (e.g., 24 and not 255.255.255.0) If unspecified, the script will also try to determine this from the routing table.

OCF_RESKEY_broadcast=Broadcast address
Broadcast address associated with the IP. If left empty, the script will determine this from the netmask.

OCF_RESKEY_iflabel=Interface label
You can specify an additional label for your IP address here. This label is appended to your interface name. If a label is specified in nic name, this parameter has no effect.

OCF_RESKEY_lvs_support=Enable support for LVS DR
Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only move it to the loopback device to allow the local node to continue to service requests, but no longer advertise it on the network.

OCF_RESKEY_mac=Cluster IP MAC address
Set the interface MAC address explicitly. Currently only used in case of the Cluster IP Alias. Leave empty to chose automatically.

OCF_RESKEY_clusterip_hash=Cluster IP hashing function
Specify the hashing algorithm used for the Cluster IP functionality.

OCF_RESKEY_arp_interval=ARP packet interval in ms
Specify the interval between unsolicited ARP packets in milliseconds.

OCF_RESKEY_arp_count=ARP packet count
Number of unsolicited ARP packets to send.

OCF_RESKEY_arp_bg=ARP from background
Whether or not to send the arp packets in the background.

OCF_RESKEY_arp_mac=ARP MAC
MAC address to send the ARP packets too. You really shouldn't be touching this.

ocf:IPaddr (7)

ocf:IPaddr — Manages virtual IPv4 addresses

Synopsis

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string]
[OCF_RESKEY_cidr_netmask=string] [OCF_RESKEY_broadcast=string]
[OCF_RESKEY_iflabel=string] [OCF_RESKEY_lvs_support=boolean]
[OCF_RESKEY_local_stop_script=string]
[OCF_RESKEY_local_start_script=string]
[OCF_RESKEY_ARP_INTERVAL_MS=integer] [OCF_RESKEY_ARP_REPEAT=integer] [OCF_RESKEY_ARP_BACKGROUND=boolean]
[OCF_RESKEY_ARP_NETMASK=string] IPaddr [start | stop | monitor | validate-all | meta-data]

Description

This script manages IP alias IP addresses It can add an IP alias, or remove one.

Supported Parameters

OCF_RESKEY_ip=IPv4 address
The IPv4 address to be configured in dotted quad notation, for example "192.168.1.1".

OCF_RESKEY_nic=Network interface
The base network interface on which the IP address will be brought online. If left empty, the script will try and determine this from the routing table. Do NOT specify an alias interface in the form eth0:1 or anything here; rather, specify the base interface only.

OCF_RESKEY_cidr_netmask=Netmask
The netmask for the interface in CIDR format. (ie, 24), or in dotted quad notation 255.255.255.0). If unspecified, the script will also try to determine this from the routing table.

OCF_RESKEY_broadcast=Broadcast address
Broadcast address associated with the IP. If left empty, the script will determine this from the netmask.

OCF_RESKEY_iflabel=Interface label
You can specify an additional label for your IP address here.

OCF_RESKEY_lvs_support=Enable support for LVS DR
Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only move it to the loopback device to allow the local node to continue to service requests, but no longer advertise it on the network.

OCF_RESKEY_local_stop_script=Script called when the IP is released
Script called when the IP is released

OCF_RESKEY_local_start_script=Script called when the IP is added
Script called when the IP is added

OCF_RESKEY_ARP_INTERVAL_MS=milliseconds between gratuitous ARPs
milliseconds between ARPs

OCF_RESKEY_ARP_REPEAT=repeat count
How many gratuitous ARPs to send out when bringing up a new address

OCF_RESKEY_ARP_BACKGROUND=run in background
run in background (no longer any reason to do this)

OCF_RESKEY_ARP_NETMASK=netmask for ARP
netmask for ARP - in nonstandard hexadecimal format.

ocf:IPsrcaddr (7)

ocf:IPsrcaddr — IPsrcaddr resource agent

Synopsis

[OCF_RESKEY_ipaddress=string] IPsrcaddr [start | stop | stop | monitor | validate-all | meta-data]

Description

Resource script for IPsrcaddr. It manages the preferred source address modification.

Supported Parameters

OCF_RESKEY_ipaddress=IP address
The IP address.

ocf:IPv6addr (7)

ocf:IPv6addr — manages IPv6 alias

Synopsis

[OCF_RESKEY_ipv6addr=string] IPv6addr [start | stop | status | monitor | validate-all | meta-data]

Description

This script manages IPv6 alias IPv6 addresses,It can add an IP6 alias, or remove one.

Supported Parameters

OCF_RESKEY_ipv6addr=IPv6 address
The IPv6 address this RA will manage

ocf:iscsi (7)

ocf:iscsi — iscsi resource agent

Synopsis

[OCF_RESKEY_portal=string] OCF_RESKEY_target=string [OCF_RESKEY_discovery_type=string] [OCF_RESKEY_iscsiadm=string] [OCF_RESKEY_udev=string] iscsi [start | stop | status | monitor | validate-all | methods | meta-data]

Description

OCF Resource Agent for iSCSI. Add (start) or remove (stop) iSCSI targets.

Supported Parameters

OCF_RESKEY_portal=portal
: The iSCSI portal address in the form: {ip_address|hostname}[":"port]

OCF_RESKEY_target=target
: The iSCSI target.

OCF_RESKEY_discovery_type=discovery_type
: Discovery type. Currently, with open-iscsi, only the sendtargets type is supported.

OCF_RESKEY_iscsiadm=iscsiadm
: iscsiadm program path.

OCF_RESKEY_udev=udev
: If the next resource depends on the udev creating a device then we wait until it is finished. On a normally loaded host this should be done quickly, but you may be unlucky. If you are not using udev set this to "no", otherwise we will spin in a loop until a timeout occurs.

ocf:Ldirectord (7)

ocf:Ldirectord — Wrapper OCF Resource Agent for ldirectord

Synopsis

`OCF_RESKEY_configfile`=string [`OCF_RESKEY_ldirectord`=string]
`Ldirectord` [start | stop | monitor | meta-data | validate-all]

Description

It's a simple OCF RA wrapper for ldirectord and uses the ldirectord interface to create the OCF compliant interface. You win monitoring of ldirectord. Be warned: Asking ldirectord status is an expensive action.

Supported Parameters

`OCF_RESKEY_configfile`=configuration file path
The full pathname of the ldirectord configuration file.

`OCF_RESKEY_ldirectord`=ldirectord binary path
The full pathname of the ldirectord.

ocf:LinuxSCSI (7)

ocf:LinuxSCSI — LinuxSCSI resource agent

Synopsis

[OCF_RESKEY_scsi=string] LinuxSCSI [start | stop | methods | status | monitor | meta-data | validate-all]

Description

This is a resource agent for LinuxSCSI. It manages the availability of a SCSI device from the point of view of the linux kernel. It make Linux believe the device has gone away, and it can make it come back again.

Supported Parameters

OCF_RESKEY_scsi=SCSI instance
The SCSI instance to be managed.

ocf:LVM (7)

ocf:LVM — LVM resource agent

Synopsis

[OCF_RESKEY_volgrpname=string] LVM [start | stop | status | monitor | methods | meta-data | validate-all]

Description

Resource script for LVM. It manages an Linux Volume Manager volume (LVM) as an HA resource.

Supported Parameters

OCF_RESKEY_volgrpname=Volume group name
The name of volume group.

ocf:MailTo (7)

ocf:MailTo — MailTo resource agent

Synopsis

[OCF_RESKEY_email=string] [OCF_RESKEY_subject=string] MailTo [start | stop | status | monitor | meta-data | validate-all]

Description

This is a resource agent for MailTo. It sends email to a sysadmin whenever a takeover occurs.

Supported Parameters

OCF_RESKEY_email=Email address
The email address of sysadmin.

OCF_RESKEY_subject=Subject
The subject of the email.

ocf:ManageRAID (7)

ocf:ManageRAID — Manages RAID devices

Synopsis

[OCF_RESKEY_raidname=string] ManageRAID [start | stop | status | monitor | validate-all | meta-data]

Description

Manages starting, stopping and monitoring of RAID devices which are preconfigured in /etc/conf.d/HB-ManageRAID.

Supported Parameters

OCF_RESKEY_raidname=RAID name

Name (case sensitive) of RAID to manage. (preconfigured in /etc/conf.d/HB-ManageRAID)

ocf:ManageVE (7)

ocf:ManageVE — OpenVZ VE resource agent

Synopsis

[OCF_RESKEY_veid=integer] ManageVE [start | stop | status | monitor | validate-all | meta-data]

Description

This OCF complaint resource agent manages OpenVZ VEs and thus requires a proper OpenVZ installation including a recent vzctl util.

Supported Parameters

OCF_RESKEY_veid=OpenVZ ID of VE

OpenVZ ID of virtual environment (see output of vzlist -a for all assigned IDs)

ocf:mysql (7)

ocf:mysql — MySQL resource agent

Synopsis

[OCF_RESKEY_binary=string] [OCF_RESKEY_config=string]
[OCF_RESKEY_datadir=string] [OCF_RESKEY_user=string]
[OCF_RESKEY_group=string] [OCF_RESKEY_log=string]
[OCF_RESKEY_pid=string] [OCF_RESKEY_socket=string]
[OCF_RESKEY_test_table=string] [OCF_RESKEY_test_user=string]
[OCF_RESKEY_test_passwd=string] [OCF_RESKEY_enable_creation=integer] [OCF_RESKEY_additional_parameters=integer] mysql [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for MySQL. It manages a MySQL Database instance as an HA resource.

Supported Parameters

OCF_RESKEY_binary=MySQL binary
Location of the MySQL binary

OCF_RESKEY_config=MySQL config
Configuration file

OCF_RESKEY_datadir=MySQL datadir
Directory containing databases

OCF_RESKEY_user=MySQL user
User running MySQL daemon

OCF_RESKEY_group=MySQL group
Group running MySQL daemon (for logfile and directory permissions)

`OCF_RESKEY_log`=MySQL log file
The logfile to be used for mysqld.

`OCF_RESKEY_pid`=MySQL pid file
The pidfile to be used for mysqld.

`OCF_RESKEY_socket`=MySQL socket
The socket to be used for mysqld.

`OCF_RESKEY_test_table`=MySQL test table
Table to be tested in monitor statement (in database.table notation)

`OCF_RESKEY_test_user`=MySQL test user
MySQL test user

`OCF_RESKEY_test_passwd`=MySQL test user password
MySQL test user password

`OCF_RESKEY_enable_creation`=Create the database if it does not exist
If the MySQL database does not exist, it will be created

`OCF_RESKEY_additional_parameters`=Additional paramters to pass to mysqld
Additional parameters which are passed to the mysqld on startup. (e.g. --skip-external-locking or --skip-grant-tables)

ocf:o2cb (7)

ocf:o2cb — OCFS2 membership layer manager.

Synopsis

OCF_RESKEY_netdev=string OCF_RESKEY_port=string [OCF_RESKEY_ocfs2_cluster=string] o2cb [start | stop | notify | monitor | validate-all | meta-data]

Description

This script manages the Oracle Cluster membership layer. It obsoletes manual configuration of the nodes in /etc/ocfs2/cluster.conf, and automates the discovery of the IP addresses uses by o2cb. It should be used below one or more ocfs2 mounts managed by Filesystem.

Supported Parameters

OCF_RESKEY_netdev=Network device for o2cb
: The network interface label which you want o3cb to run over.

OCF_RESKEY_port=Port number
: The port number you want o2cb to use for communications.

OCF_RESKEY_ocfs2_cluster=o2cb cluster name
: The name of the cluster for which this resource is managing the membership. The default is likely fine.

ocf:oracle (7)

ocf:oracle — oracle resource agent

Synopsis

OCF_RESKEY_sid=string [OCF_RESKEY_home=string] [OCF_RESKEY_user=string] [OCF_RESKEY_ipcrm=string] oracle [start | stop | status | monitor | validate-all | methods | meta-data]

Description

Resource script for oracle. Manages an Oracle Database instance as an HA resource.

Supported Parameters

OCF_RESKEY_sid=sid
The Oracle SID (aka ORACLE_SID).

OCF_RESKEY_home=home
The Oracle home directory (aka ORACLE_HOME). If not specified, then the SID along with its home should be listed in /etc/oratab.

OCF_RESKEY_user=user
The Oracle owner (aka ORACLE_OWNER). If not specified, then it is set to the owner of file $ORACLE_HOME/dbs/*${ORACLE_SID}.ora. If this does not work for you, just set it explicitely.

OCF_RESKEY_ipcrm=ipcrm
Sometimes IPC objects (shared memory segments and semaphores) belonging to an Oracle instance might be left behind which prevents the instance from starting. It is not easy to figure out which shared segments belong to which instance, in particular when more instances are running as same user. What we use here is the "oradebug" feature and its "ipc" trace utility. It is not optimal to parse the debugging information, but I am not aware of any other way to find out about the IPC infor-

mation. In case the format or wording of the trace report changes, parsing might fail. There are some precautions, however, to prevent stepping on other peoples toes. There is also a dumpinstipc option which will make us print the IPC objects which belong to the instance. Use it to see if we parse the trace file correctly. Three settings are possible: - none: don't mess with IPC and hope for the best (beware: you'll probably be out of luck, sooner or later) - instance: try to figure out the IPC stuff which belongs to the instance and remove only those (default; should be safe) - orauser: remove all IPC belonging to the user which runs the instance (don't use this if you run more than one instance as same user or if other apps running as this user use IPC) The default setting "instance" should be safe to use, but in that case we cannot guarantee that the instance will start. In case IPC objects were already left around, because, for instance, someone mercilessly killing Oracle processes, there is no way any more to find out which IPC objects should be removed. In that case, human intervention is necessary, and probably _all_ instances running as same user will have to be stopped. The third setting, "orauser", guarantees IPC objects removal, but it does that based only on IPC objects ownership, so you should use that only if every instance runs as separate user. Please report any problems. Suggestions/fixes welcome.

ocf:oralsnr (7)

ocf:oralsnr — oralsnr resource agent

Synopsis

OCF_RESKEY_sid=string [OCF_RESKEY_home=string]
[OCF_RESKEY_user=string] OCF_RESKEY_listener=string oralsnr [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for Oracle Listener. It manages an Oracle Listener instance as an HA resource.

Supported Parameters

OCF_RESKEY_sid=sid
The Oracle SID (aka ORACLE_SID). Necessary for the monitor op, i.e. to do tnsping SID.

OCF_RESKEY_home=home
The Oracle home directory (aka ORACLE_HOME). If not specified, then the SID should be listed in /etc/oratab.

OCF_RESKEY_user=user
Run the listener as this user.

OCF_RESKEY_listener=listener
Listener instance to be started (as defined in listener.ora). Defaults to LISTENER.

ocf:pgsql (7)

ocf:pgsql — pgsql resource agent

Synopsis

[OCF_RESKEY_pgctl=string] [OCF_RESKEY_start_opt=string]
[OCF_RESKEY_ctl_opt=string] [OCF_RESKEY_psql=string]
[OCF_RESKEY_pgdata=string] [OCF_RESKEY_pgdba=string]
[OCF_RESKEY_pghost=string] [OCF_RESKEY_pgport=string]
[OCF_RESKEY_pgdb=string] [OCF_RESKEY_logfile=string]
[OCF_RESKEY_stop_escalate=string] pgsql [start | stop | status | monitor | meta-data | validate-all | methods]

Description

Resource script for PostgreSQL. It manages a PostgreSQL as an HA resource.

Supported Parameters

OCF_RESKEY_pgctl=pgctl
Path to pg_ctl command.

OCF_RESKEY_start_opt=start_opt
Start options (-o start_opt in pgi_ctl). "-i -p 5432" for example.

OCF_RESKEY_ctl_opt=ctl_opt
Additional pg_ctl options (-w, -W etc..). Default is ""

OCF_RESKEY_psql=psql
Path to psql command.

OCF_RESKEY_pgdata=pgdata
Path PostgreSQL data directory.

`OCF_RESKEY_pgdba`=pgdba
User that owns PostgreSQL.

`OCF_RESKEY_pghost`=pghost
Hostname/IP Addreess where PosrgeSQL is listening

`OCF_RESKEY_pgport`=pgport
Port where PosrgeSQL is listening

`OCF_RESKEY_pgdb`=pgdb
Database that will be used for monitoring.

`OCF_RESKEY_logfile`=logfile
Path to PostgreSQL server log output file.

`OCF_RESKEY_stop_escalate`=stop escalation
Number of retries (using -m fast) before resorting to -m immediate

ocf:pingd (7)

ocf:pingd — pingd resource agent

Synopsis

[OCF_RESKEY_pidfile=string] [OCF_RESKEY_user=string]
[OCF_RESKEY_dampen=integer] [OCF_RESKEY_set=integer]
[OCF_RESKEY_name=integer] [OCF_RESKEY_section=integer]
[OCF_RESKEY_multiplier=integer] [OCF_RESKEY_host_list=integer]
pingd [start | stop | monitor | meta-data | validate-all]

Description

This is a pingd Resource Agent. It records (in the CIB) the current number of ping nodes a node can connect to.

Supported Parameters

OCF_RESKEY_pidfile=PID file
PID file

OCF_RESKEY_user=The user we want to run pingd as
The user we want to run pingd as

OCF_RESKEY_dampen=Dampening interval
The time to wait (dampening) further changes occur

OCF_RESKEY_set=Set name
The name of the instance_attributes set to place the value in. Rarely needs to be specified.

OCF_RESKEY_name=Attribute name
The name of the attributes to set. This is the name to be used in the constraints.

`OCF_RESKEY_section`=Section name
: The section place the value in. Rarely needs to be specified.

`OCF_RESKEY_multiplier`=Value multiplier
: The number by which to multiply the number of connected ping nodes by

`OCF_RESKEY_host_list`=Host list
: The list of ping nodes to count. Defaults to all configured ping nodes. Rarely needs to be specified.

ocf:portblock (7)

ocf:portblock — portblock resource agent

Synopsis

[OCF_RESKEY_protocol=string] [OCF_RESKEY_portno=integer] [OCF_RESKEY_action=string] portblock [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for portblock. It is used to temporarily block ports using iptables.

Supported Parameters

OCF_RESKEY_protocol=protocol
The protocol used to be blocked/unblocked.

OCF_RESKEY_portno=portno
The port number used to be blocked/unblocked.

OCF_RESKEY_action=action
The action (block/unblock) to be done on the protocol::portno.

ocf:Pure-FTPd (7)

ocf:Pure-FTPd — OCF Resource Agent compliant FTP script.

Synopsis

OCF_RESKEY_script=string OCF_RESKEY_conffile=string
OCF_RESKEY_daemon_type=string [OCF_RESKEY_pidfile=string]
Pure-FTPd [start | stop | monitor | validate-all | meta-data]

Description

This script manages Pure-FTPd in an Active-Passive setup

Supported Parameters

OCF_RESKEY_script=Script name with full path
The full path to the Pure-FTPd startup script. For example, "/sbin/pure-config.pl"

OCF_RESKEY_conffile=Configuration file name with full path
The Pure-FTPd configuration file name with full path. For example, "/etc/pure-ftpd/pure-ftpd.conf"

OCF_RESKEY_daemon_type=Configuration file name with full path
The Pure-FTPd daemon to be called by pure-ftpd-wrapper. Valid options are "" for pure-ftpd, "mysql" for pure-ftpd-mysql, "postgresql" for pure-ftpd-postgresql and "ldap" for pure-ftpd-ldap

OCF_RESKEY_pidfile=PID file
PID file

ocf:Raid1 (7)

ocf:Raid1 — RAID1 resource agent

Synopsis

[OCF_RESKEY_raidconf=string] [OCF_RESKEY_raiddev=string] [OCF_RESKEY_homehost=string] Raid1 [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for RAID1. It manages a software Raid1 device on a shared storage medium.

Supported Parameters

OCF_RESKEY_raidconf=RAID config file
The RAID configuration file. e.g. /etc/raidtab or /etc/mdadm.conf.

OCF_RESKEY_raiddev=block device
The block device to use.

OCF_RESKEY_homehost=Homehost for mdadm
The value for the homehost directive; this is an mdadm feature to protect RAIDs against being activated by accident. It is recommended to create RAIDs managed by the cluster with "homehost" set to a special value, so they are not accidentially auto-assembled by nodes not supposed to own them.

ocf:rsyncd (7)

ocf:rsyncd — OCF Resource Agent compliant rsync daemon script.

Synopsis

[OCF_RESKEY_binpath=string] [OCF_RESKEY_conffile=string] [OCF_RESKEY_bwlimit=string] rsyncd [start | stop | monitor | validate-all | meta-data]

Description

This script manages rsync daemon

Supported Parameters

OCF_RESKEY_binpath=Full path to the rsync binary
The rsync binary path. For example, "/usr/bin/rsync"

OCF_RESKEY_conffile=Configuration file name with full path
The rsync daemon configuration file name with full path. For example, "/etc/rsyncd.conf"

OCF_RESKEY_bwlimit=limit I/O bandwidth, KBytes per second
This option allows you to specify a maximum transfer rate in kilobytes per second. This option is most effective when using rsync with large files (several megabytes and up). Due to the nature of rsync transfers, blocks of data are sent, then if rsync determines the transfer was too fast, it will wait before sending the next data block. The result is an average transfer rate equaling the specified limit. A value of zero specifies no limit.

ocf:SAPDatabase (7)

ocf:SAPDatabase — SAP database resource agent

Synopsis

OCF_RESKEY_SID=string OCF_RESKEY_DIR_EXECUTABLE=string
OCF_RESKEY_DBTYPE=string OCF_RESKEY_NETSERVICENAME=string
OCF_RESKEY_DBJ2EE_ONLY=boolean OCF_RESKEY_JAVA_HOME=string
OCF_RESKEY_STRICT_MONITORING=boolean
OCF_RESKEY_AUTOMATIC_RECOVER=boolean
OCF_RESKEY_DIR_BOOTSTRAP=string OCF_RESKEY_DIR_SECSTORE=string
OCF_RESKEY_PRE_START_USEREXIT=string
OCF_RESKEY_POST_START_USEREXIT=string
OCF_RESKEY_PRE_STOP_USEREXIT=string
OCF_RESKEY_POST_STOP_USEREXIT=string SAPDatabase [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for SAP databases. It manages a SAP database of any type as an HA resource.

Supported Parameters

OCF_RESKEY_SID=SAP system ID
The unique SAP system identifier. e.g. P01

OCF_RESKEY_DIR_EXECUTABLE=path of sapstartsrv and sapcontrol
The full qualified path where to find sapstartsrv and sapcontrol.

OCF_RESKEY_DBTYPE=database vendor
The name of the database vendor you use. Set either: ORA,DB6,ADA

OCF_RESKEY_NETSERVICENAME=listener name
The Oracle TNS listener name.

OCF_RESKEY_DBJ2EE_ONLY=only JAVA stack installed
If you do not have a ABAP stack installed in the SAP database, set this to TRUE

OCF_RESKEY_JAVA_HOME=Path to Java SDK
This is only needed if the DBJ2EE_ONLY parameter is set to true. Enter the path to the Java SDK which is used by the SAP WebAS Java

OCF_RESKEY_STRICT_MONITORING=Activates application level monitoring
This controls how the resource agent monitors the database. If set to true, it will use SAP tools to test the connect to the database. Do not use with Oracle, because it will result in unwanted failovers in case of an archiver stuck

OCF_RESKEY_AUTOMATIC_RECOVER=Enable or disable automatic startup recovery
The SAPDatabase resource agent tries to recover a failed start attempt automaticaly one time. This is done by running a forced abort of the RDBMS and/or executing recovery commands.

OCF_RESKEY_DIR_BOOTSTRAP=path to j2ee bootstrap directory
The full qualified path where to find the J2EE instance bootstrap directory. e.g. /usr/sap/P01/J00/j2ee/cluster/bootstrap

OCF_RESKEY_DIR_SECSTORE=path to j2ee secure store directory
The full qualified path where to find the J2EE security store directory. e.g. /usr/sap/P01/SYS/global/security/lib/tools

OCF_RESKEY_PRE_START_USEREXIT=path to a pre-start script
The full qualified path where to find a script or program which should be executed before this resource gets started.

OCF_RESKEY_POST_START_USEREXIT=path to a post-start script
The full qualified path where to find a script or program which should be executed after this resource got started.

OCF_RESKEY_PRE_STOP_USEREXIT=path to a pre-start script
The full qualified path where to find a script or program which should be executed before this resource gets stopped.

`OCF_RESKEY_POST_STOP_USEREXIT`=path to a post-start script

The full qualified path where to find a script or program which should be executed after this resource got stopped.

ocf:SAPInstance (7)

ocf:SAPInstance — SAP instance resource agent

Synopsis

OCF_RESKEY_InstanceName=string OCF_RESKEY_DIR_EXECUTABLE=string
OCF_RESKEY_DIR_PROFILE=string OCF_RESKEY_START_PROFILE=string
OCF_RESKEY_START_WAITTIME=string
OCF_RESKEY_AUTOMATIC_RECOVER=boolean
OCF_RESKEY_PRE_START_USEREXIT=string
OCF_RESKEY_POST_START_USEREXIT=string
OCF_RESKEY_PRE_STOP_USEREXIT=string
OCF_RESKEY_POST_STOP_USEREXIT=string SAPInstance [start | recover | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for SAP. It manages a SAP Instance as an HA resource.

Supported Parameters

OCF_RESKEY_InstanceName=instance name: SID_INSTANCE_VIR-HOSTNAME
The full qualified SAP instance name. e.g. P01_DVEBMGS00_sapp01ci

OCF_RESKEY_DIR_EXECUTABLE=path of sapstartsrv and sapcontrol
The full qualified path where to find sapstartsrv and sapcontrol.

OCF_RESKEY_DIR_PROFILE=path of start profile
The full qualified path where to find the SAP START profile.

OCF_RESKEY_START_PROFILE=start profile name
The name of the SAP START profile.

OCF_RESKEY_START_WAITTIME=Check the successful start after that time (do not wait for J2EE-Addin)

After that time in seconds a monitor operation is executed by the resource agent. Does the monitor return SUCCESS, the start is handled as SUCCESS. This is useful to resolve timing problems with e.g. the J2EE-Addin instance.

OCF_RESKEY_AUTOMATIC_RECOVER=Enable or disable automatic startup recovery

The SAPInstance resource agent tries to recover a failed start attempt automaticaly one time. This is done by killing runing instance processes and executing cleanipc.

OCF_RESKEY_PRE_START_USEREXIT=path to a pre-start script

The full qualified path where to find a script or program which should be executed before this resource gets started.

OCF_RESKEY_POST_START_USEREXIT=path to a post-start script

The full qualified path where to find a script or program which should be executed after this resource got started.

OCF_RESKEY_PRE_STOP_USEREXIT=path to a pre-start script

The full qualified path where to find a script or program which should be executed before this resource gets stopped.

OCF_RESKEY_POST_STOP_USEREXIT=path to a post-start script

The full qualified path where to find a script or program which should be executed after this resource got stopped.

ocf:SendArp (7)

ocf:SendArp — SendArp resource agent

Synopsis

[OCF_RESKEY_ip=string] [OCF_RESKEY_nic=string] SendArp [start | stop | monitor | meta-data | validate-all]

Description

This script send out gratuitous Arp for an IP address

Supported Parameters

OCF_RESKEY_ip=IP address
The IP address for sending arp package.

OCF_RESKEY_nic=NIC
The nic for sending arp package.

ocf:ServeRAID (7)

ocf:ServeRAID — ServeRAID resource agent

Synopsis

[OCF_RESKEY_serveraid=integer] [OCF_RESKEY_mergegroup=integer]
ServeRAID [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for ServeRAID. It enables/disables shared ServeRAID merge groups.

Supported Parameters

OCF_RESKEY_serveraid=serveraid
The adapter number of the ServeRAID adapter.

OCF_RESKEY_mergegroup=mergegroup
The logical drive under consideration.

ocf:SphinxSearchDaemon (7)

ocf:SphinxSearchDaemon — searchd resource agent

Synopsis

OCF_RESKEY_config=string [OCF_RESKEY_searchd=string]
[OCF_RESKEY_search=string] [OCF_RESKEY_testQuery=string]
SphinxSearchDaemon [start | stop | monitor | meta-data | validate-all]

Description

This is a searchd Resource Agent. It manages the Sphinx Search Daemon.

Supported Parameters

OCF_RESKEY_config=Configuration file
searchd configuration file

OCF_RESKEY_searchd=searchd binary
searchd binary

OCF_RESKEY_search=search binary
Search binary for functional testing in the monitor action.

OCF_RESKEY_testQuery=test query
Test query for functional testing in the monitor action. The query does not need to match any documents in the index. The purpose is merely to test whether the search daemon is is able to query its indices and respond properly.

ocf:Stateful (7)

ocf:Stateful — Example stateful resource agent

Synopsis

`OCF_RESKEY_state`=string `Stateful` [start | stop | monitor | meta-data | validate-all]

Description

This is an example resource agent that impliments two states

Supported Parameters

`OCF_RESKEY_state`=State file
Location to store the resource state in

ocf:SysInfo (7)

ocf:SysInfo — SysInfo resource agent

Synopsis

[OCF_RESKEY_pidfile=string] [OCF_RESKEY_delay=string] SysInfo [start | stop | monitor | meta-data | validate-all]

Description

This is a SysInfo Resource Agent. It records (in the CIB) various attributes of a node Sample Linux output: arch: i686 os: Linux-2.4.26-gentoo-r14 free_swap: 1999 cpu_info: Intel(R) Celeron(R) CPU 2.40GHz cpu_speed: 4771.02 cpu_cores: 1 cpu_load: 0.00 ram_total: 513 ram_free: 117 root_free: 2.4 Sample Darwin output: arch: i386 os: Darwin-8.6.2 cpu_info: Intel Core Duo cpu_speed: 2.16 cpu_cores: 2 cpu_load: 0.18 ram_total: 2016 ram_free: 787 root_free: 13 Units: free_swap: Mb ram_*: Mb root_free: Gb cpu_speed (Linux): bogomips cpu_speed (Darwin): Ghz

Supported Parameters

OCF_RESKEY_pidfile=PID file
: PID file

OCF_RESKEY_delay=Dampening Delay
: Interval to allow values to stabilize

ocf:tomcat (7)

ocf:tomcat — tomcat resource agent

Synopsis

OCF_RESKEY_tomcat_name=string OCF_RESKEY_script_log=string
[OCF_RESKEY_tomcat_stop_timeout=integer]
[OCF_RESKEY_tomcat_suspend_trialcount=integer]
[OCF_RESKEY_tomcat_user=string] [OCF_RESKEY_statusurl=string]
OCF_RESKEY_java_home=string OCF_RESKEY_catalina_home=string
OCF_RESKEY_catalina_pid=string tomcat [start | stop | status | monitor | meta-data | validate-all]

Description

Resource script for tomcat. It manages a Tomcat instance as an HA resource.

Supported Parameters

OCF_RESKEY_tomcat_name=The name of the resource
The name of the resource

OCF_RESKEY_script_log=A destination of the log of this script
A destination of the log of this script

OCF_RESKEY_tomcat_stop_timeout=Time-out at the time of the stop
Time-out at the time of the stop

OCF_RESKEY_tomcat_suspend_trialcount=The re-try number of times awaiting a stop
The re-try number of times awaiting a stop

OCF_RESKEY_tomcat_user=A user name to start a resource
A user name to start a resource

`OCF_RESKEY_statusurl`=URL for state confirmation
URL for state confirmation

`OCF_RESKEY_java_home`=Home directory of the Java
Home directory of the Java

`OCF_RESKEY_catalina_home`=Home directory of Tomcat
Home directory of Tomcat

`OCF_RESKEY_catalina_pid`=A PID file name of Tomcat
A PID file name of Tomcat

ocf:VIPArip (7)

ocf:VIPArip — Virtual IP Address by RIP2 protocol

Synopsis

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string] VIPArip [start | stop | monitor | validate-all | meta-data]

Description

Virtual IP Address by RIP2 protocol. This script manages IP alias in different subnet with quagga/ripd. It can add an IP alias, or remove one.

Supported Parameters

OCF_RESKEY_ip=The IP address in different subnet
The IPv4 address in different subnet, for example "192.168.1.1".

OCF_RESKEY_nic=The nic for broadcast the route information
The nic for broadcast the route information. The ripd uses this nic to broadcast the route informaton to others

ocf:WAS6 (7)

ocf:WAS6 — WAS6 resource agent

Synopsis

[OCF_RESKEY_profile=string] WAS6 [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for WAS6. It manages a Websphere Application Server (WAS6) as an HA resource.

Supported Parameters

OCF_RESKEY_profile=profile name
The WAS profile name.

ocf:WAS (7)

ocf:WAS — WAS resource agent

Synopsis

[OCF_RESKEY_config=string] [OCF_RESKEY_port=integer] WAS [start | stop | status | monitor | validate-all | meta-data | methods]

Description

Resource script for WAS. It manages a Websphere Application Server (WAS) as an HA resource.

Supported Parameters

OCF_RESKEY_config=configration file
The WAS-configuration file.

OCF_RESKEY_port=port
The WAS-(snoop)-port-number.

ocf:WinPopup (7)

ocf:WinPopup — WinPopup resource agent

Synopsis

[OCF_RESKEY_hostfile=string] WinPopup [start | stop | status | monitor | validate-all | meta-data]

Description

Resource script for WinPopup. It sends WinPopups message to a sysadmin's workstation whenever a takeover occurs.

Supported Parameters

OCF_RESKEY_hostfile=Host file

The file containing the hosts to send WinPopup messages to.

ocf:Xen (7)

ocf:Xen — Manages Xen DomUs

Synopsis

[OCF_RESKEY_xmfile=string] [OCF_RESKEY_allow_migrate=boolean]
[OCF_RESKEY_allow_mem_management=boolean]
[OCF_RESKEY_reserved_Dom0_memory=string]
[OCF_RESKEY_monitor_scripts=string] Xen [start | stop | migrate_from | migrate_to | status | monitor | meta-data | validate-all]

Description

Resource Agent for the Xen Hypervisor. Manages Xen virtual machine instances by mapping cluster resource start and stop, to Xen create and shutdown, respectively. Paravirtualized guests can also be migrated by enabling the meta_attribute allow_migrate.

Supported Parameters

OCF_RESKEY_xmfile=Xen control file
Absolute path to the Xen control file, for this virtual machine.

OCF_RESKEY_allow_migrate=Use live migration
This bool parameters allows to use live migration for paravirtual machines.

OCF_RESKEY_allow_mem_management=Use dynamic memory management
This parameter enables dynamic adjustment of memory for start and stop actions used for Dom0 and the DomUs. The default is to not adjust memory dynamically.

OCF_RESKEY_reserved_Dom0_memory=Minimum Dom0 memory
In case memory management is used, this parameter defines the minimum amount of memory to be reserved for the dom0. The default minimum memory is 512MB.

`OCF_RESKEY_monitor_scripts`=list of space separated monitor scripts

To additionally monitor services within the unprivileged domain, add this parameter with a list of scripts to monitor. NB: In this case make sure to set the start-delay of the monitor operation to at least the time it takes for the DomU to start all services.

ocf:Xinetd (7)

ocf:Xinetd — Xinetd resource agent

Synopsis

[OCF_RESKEY_service=string] Xinetd [start | stop | restart | status | monitor | validate-all | meta-data]

Description

Resource script for Xinetd. It starts/stops services managed by xinetd. Note that the xinetd daemon itself must be running: we are not going to start it or stop it ourselves. Important: in case the services managed by the cluster are the only ones enabled, you should specify the -stayalive option for xinetd or it will exit on Heartbeat stop. Alternatively, you may enable some internal service such as echo.

Supported Parameters

OCF_RESKEY_service=service name
: The service name managed by xinetd.

Terminology

cluster
: A high-performance cluster is a group of computers (real or virtual) sharing application load to get things done fast. A high availability cluster is designed primarily to secure the highest possible availability of services.

cluster partition
: Whenever communication fails between one or more nodes and the rest of the cluster, a cluster partition occurs. The nodes of a cluster partition are still active and able to communicate with each other, but they are unaware of the nodes with which they cannot communicate. As the loss of the other partition cannot be confirmed, a split brain scenario develops (see also split brain (page 178)).

consensus cluster membership (CCM)
: The CCM determines which nodes make up the cluster and shares this information across the cluster. Any new addition and any loss of nodes or quorum is delivered by the CCM. A CCM module runs on each node of the cluster.

cluster information base (CIB)
: A representation of the whole cluster configuration and status (node membership, resources, constraints, etc.) written in XML and residing in memory. A master CIB is kept and maintained on the DC and replicated to the other nodes.

cluster resource manager (CRM)
: The main management entity responsible for coordinating all nonlocal interactions. Each node of the cluster has its own CRM, but the one running on the DC is the one elected to relay decisions to the other nonlocal CRMs and process their input. A CRM interacts with a number of components: local resource managers both on its own node and on the other nodes, nonlocal CRMs, administrative commands, the fencing functionality, and the membership layer.

designated coordinator (DC)
: The “master” node. This node is where the master copy of the CIB is kept. All other nodes get their configuration and resource allocation information from the current DC. The DC is elected from all nodes in the cluster after a membership change.

Distributed replicated block device (drbd)
: DRBD is a block device designed for building high availability clusters. The whole block device is mirrored via a dedicated network and is seen as a network RAID-1.

failover
: Occurs when a resource or node fails on one machine and the affected resources are started on another node.

fencing
: Describes the concept of preventing access to a shared resource by non-cluster members. It can be archived by killing (shutting down) a "misbehaving" node to prevent it from causing trouble, locking resources away from a node whose status is uncertain, or in several other ways. Furthermore fencing is distinguished between node and resource fencing.

Heartbeat resource agent
: Heartbeat resource agents were widely used with Heartbeat version 1. Their use is deprecated, but still supported in version 2. A Heartbeat resource agent can perform `start`, `stop`, and `status` operations and resides under `/etc/ha.d/resource.d` or `/etc/init.d`. For more information about Heartbeat resource agents, refer to `http://www.linux-ha.org/HeartbeatResourceAgent`.

local resource manager (LRM)
: The local resource manager (LRM) is responsible for performing operations on resources. It uses the resource agent scripts to carry out the work. The LRM is "dumb" in that it does not know of any policy by itself. It needs the DC to tell it what to do.

LSB resource agent
: LSB resource agents are standard LSB init scripts. LSB init scripts are not limited to use in a high availability context. Any LSB-compliant Linux system uses LSB init scripts to control services. Any LSB resource agent supports a `start`, `stop`, `restart`, `status` and `force-reload` option and may optionally provide `try-restart` and `reload` as well. LSB resource agents are located in `/etc/init.d`. Find more information about LSB resource agents and the actual specification at `http://www.linux-ha.org/LSBResourceAgent` and `http://www.linux-foundation.org/spec/refspecs/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html`

node
: Any computer (real or virtual) that is a member of a cluster and invisible for the user.

pingd
: The ping daemon. It continuously contacts one or more servers outside the cluster with ICMP pings.

policy engine (PE)
: The policy engine computes the actions that need to be taken to implement policy changes in the CIB. This information is then passed on to the transaction engine, which in turn implements the policy changes in the cluster setup. The PE always runs on the DC.

OCF resource agent
: OCF resource agents are similar to LSB resource agents (init scripts). Any OCF resource agent must support `start`, `stop`, and `status` (sometimes called `monitor`) options. Additionally, they support a `metadata` option that returns the description of the resource agent type in XML. Additional options may be supported, but are not mandatory. OCF resource agents reside in `/usr/lib/ocf/resource.d/`*`provider`*`.` Find more information about OCF resource agents and a draft of the specification at `http://www.linux-ha.org/OCFResourceAgent` and `http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD`.

quorum
: In a cluster, a cluster partition is defined to have quorum (is "quorate") if it has the majority of nodes (or votes). Quorum distinguishes exactly one partition. It is part of the algorithm to prevent several disconnected partitions or nodes from proceeding and causing data and service corruption (split brain). Quorum is a prerequisite for fencing, which then ensures that quorum is indeed unique.

resource
: Any type of service or application that is known to Heartbeat. Examples include an IP address, a file system, or a database.

resource agent (RA)
: A resource agent (RA) is a script acting as a proxy to manage a resource. There are three different kinds of resource agents: OCF (Open Cluster Framework) re-

source agents, LSB resource agents (Standard LSB init scripts), and Heartbeat resource agents (Heartbeat v1 resources).

Single Point of Failure (SPOF)
: A single point of failure (SPOF) is any component of a cluster that, should it fail, triggers the failure of the entire cluster.

split brain
: A scenario in which the cluster nodes are divided into two or more groups that do not know of each other (either through a software or hardware failure). To prevent a split brain situation from badly affecting the entire cluster, STONITH must come to the rescue. Also known as a "partitioned cluster" scenario.

STONITH
: The acronym for "Shoot the other node in the head" which is basically bringing down a misbehaving node to prevent it from causing trouble in the cluster.

transition engine (TE)
: The transition engine (TE) takes the policy directives from the PE and carries them out. The TE always runs on the DC. From there, it instructs the local resource managers on the others nodes which actions to take.